jQuery

A Biblioteca do Programador JavaScript

Aprenda a criar efeitos de alto impacto em seu site com a biblioteca JavaScript mais utilizada pelos desenvolvedores web

Maurício Samy Silva
www.maujor.com

novatec

Sumário

Agradecimentos 11

Isenção de responsabilidade 12

Sobre o autor 13

Introdução 15

Parte I ■ Fundamentos teóricos da biblioteca jQuery 23

Capítulo 1 ■ O que é jQuery? 25

1.1 Definições e conceitos 25
1.1.1 O que é jQuery? 25
1.1.2 Para que serve jQuery? 27
1.1.3 jQuery em conformidade com os Padrões Web 28
1.1.4 Características da biblioteca jQuery 29
1.1.5 Como instalar jQuery 30
1.1.6 jQuery na prática 32
1.1.7 Fundamentos jQuery 40

Capítulo 2 ■ Funções-padrão e seletores jQuery 45

2.1 Funções-padrão jQuery 45
2.2 Seletores jQuery 55
2.2.1 Seletores simples 57
2.2.2 Seletores compostos 60
2.2.3 Seletores – Filtros básicos 64
2.2.4 Seletores de conteúdo 73
2.2.5 Seletores de visibilidade 76
2.2.6 Seletores de atributo 79
2.2.7 Filtros para seletores-filho 85
2.2.8 Seletores para formulários 88
2.2.9 Filtros para formulários 97

Capítulo 3 ■ Métodos de manipulação do DOM 101

3.1 Manipulação de atributos gerais 101
3.2 Manipulação do atributo class 105
3.3 Manipulação de conteúdos html 108

3.4 Manipulação de textos 109
3.5 Manipulação de valores 111
3.6 Manipulação de conteúdos 113

Capítulo 4 ■ CSS e inspeção do DOM 123
4.1 Estilização geral 123
4.2 Posicionamento 125
4.3 Largura e altura 130
4.4 Inspeção do DOM 136

Capítulo 5 ■ Eventos 151
5.1 Eventos auxiliares 151
5.2 Eventos de interação 164
5.3 Manipuladores de eventos 166
5.4 Notas sobre eventos 169

Capítulo 6 ■ Efeitos 173
6.1 Efeitos básicos 173
6.2 Efeitos corrediços 177
6.3 Efeitos de opacidade 179
6.4 Efeitos personalizados 181

Capítulo 7 ■ Funções utilitárias 185
7.1 Introdução 185
7.2 Flags para agentes de usuário 186
7.3 Operações com arrays e objetos 190
7.4 Teste de função 196
7.5 Operação com string 197
7.6 Funções utilitárias personalizadas 198
7.6.1 Sintaxe geral 198
7.6.2 Funçao `$.corTexto` 200

Parte II ■ jQuery na prática 211

Capítulo 8 ■ Animações básicas 213
8.1 Marcação mínima 213
8.2 Animação com `show()` e `hide()` 215
8.3 Animação com `toggle()` 219
8.4 Suavizando a animação 228
8.5 Animação com `slideUp()` e `slideDown()` 231
8.6 Animação com `fadeIn()` e `fadeOut()` 233
8.7 Animação personalizada com `animate()` 234

Capítulo 9 ■ Revelando conteúdos 239
9.1 FAQ CSS 239
Marcação básica 240

Primeira etapa 243
Segunda etapa 248
Terceira etapa 254
Quarta etapa 259
Quinta etapa 261
9.2 Página de notícias 265
Marcação básica 266
Primeira etapa 268
Segunda etapa 271

Capítulo 10 ■ Efeitos em tabelas 275

10.1 Destinação das tabelas HTML 275
10.2 Marcação de tabelas 275
10.3 Tabela de horários de ônibus 276
Marcação básica 276
10.4 Efeito zebra 280
Zebra par-ímpar 281
Zebra dois-dois 283
Progressão aritmética 286
Zebra três cores 286
Zebra três-três 288
10.5 Efeitos para destacar 290
Destacar linhas 291
Destacar colunas 292
Destacar linhas seletivamente 295
Revelar e esconder linhas 300
Advertência 305

Capítulo 11 ■ Efeitos em formulários 307

11.1 Validação de formulários 307
11.2 Placeholder para campos 307
11.3 Dicas de preenchimento 311
11.4 Desabilitar campos 312
11.5 Revelar campos 316
11.6 Elemento legend 318
11.7 Selecionar todos 320
11.8 Validar 322

Capítulo 12 ■ Imagens 325

12.1 Introdução 325
12.1.1 Imagens acessíveis 326
12.2 Ampliação de imagens 327
12.3 Galerias de imagens 340
12.3.1 Galeria com thumbnails 342
12.4 Slide-show 347

Capítulo 13 ■ Plug-ins 355

13.1 Introdução 355
13.1.1 Plug-ins de terceiros 355
13.1.2 Plug-ins nativos 356
13.2 Plug-in jCarousel 356
13.2.1 Instalação 358
13.2.2 Nome das imagens 359
13.3 Carrosséis horizontal e vertical 360
13.4 Carrossel com scroll automático 362
13.5 Carrossel com comandos externos 365
13.6 Carrossel com efeito thickbox 368
13.7 Plug-in jQuery Accordion 370
13.7.1 Instalação 371
13.7.2 Folhas de estilo 372
13.8 Sanfona simples 372
13.9 Menu sanfona 375

Capítulo 14 ■ Menu Maujor 379

14.1 Introdução 379
14.2 HTML e CSS 380
14.3 Script 387

Apêndice A ■ Seletores 389

A.1 Seletor tipo 389
A.2 Seletor identificador único 390
A.3 Seletor classe 390
A.3.1 Classificação dos seletores 390
A.4 Seletores avançados 396

Apêndice B ■ Codificação de caracteres para HTML 397

Caracteres especiais para HTML 397
Caracteres matemáticos, gregos e símbolos para HTML 398
Caracteres para HTML – ISO-8859-1 399

Apêndice C ■ Elementos HTML 401

apêndice D ■ Atributos HTML 405

apêndice E ■ FAQ jQuery 415

Referências bibliográficas 425

Índice remissivo 427

Agradecimentos

Agradeço a Deus por ter me dado forças, disposição e motivação para escrever este livro.

Sou grato aos profissionais da Novatec Editora, diretamente envolvidos na publicação do livro, em particular ao editor Rubens Prates, que, ao longo de todo o processo de criação, esteve presente guiando-me com suas dicas e informações sobre as particularidades e implicações editoriais próprias à fase de criação de um livro.

Meu maior agradecimento é a você, leitor, por interessar-se em aprender jQuery e honrar-me com a leitura deste livro.

Isenção de responsabilidade

Todos os esforços foram feitos na elaboração deste livro para assegurar o fornecimento de informações as mais precisas, completas e exatas. Contudo, as informações aqui contidas são fornecidas "como estão" e sem nenhuma garantia, seja expressa, seja implícita. O autor, a editora, os distribuidores e qualquer entidade envolvida direta ou indiretamente na sua comercialização não assumirão nenhuma responsabilidade por qualquer prejuízo ou dano, direto ou indireto, consequente às informações contidas neste livro.

Sobre o autor

Maurício Samy Silva é graduado em Engenharia Civil pelo Instituto Militar de Engenharia (IME). Fundador e ex-diretor-presidente da Planep Engenharia, exerceu o magistério paralelamente à Engenharia, tendo sido, ao longo de 25 anos, professor de Geometria Descritiva e Matemática. Seu primeiro contato com programação para computadores deu-se ainda quando aluno, ao aprender a linguagem Fortran. Àquela época, era comum passarem-se noites trabalhando em uma máquina perfuradora de cartões que seriam processados em um então poderoso IBM/360. Em 1982, adquiriu seu primeiro computador pessoal, um TK-80 com processador Z-80A de 3,25MHz, teclado de membrana com 40 teclas e memória RAM de 1KB expansível até 16KB. Trabalhou com os modelos TK-82, TK-85, TK-2000, DGT-100 e assim por diante, até os dias atuais.

Sua experiência com desenvolvimento de sites iniciou-se em 1999, com o FrontPage, considerada uma ferramenta sensacional na construção de sites. Em 2002, por acaso teve seu primeiro contato com o site do W3C e, como ficou vivamente impressionado com a proposta desse consórcio, começou a pesquisar e a estudar as Web Standards, tendo como fonte de consulta o material publicado na internet em língua inglesa. Ao contrário do que ocorre nos dias atuais, quando vários desenvolvedores escrevem em blogs pessoais sobre Web Standards, a literatura a respeito do assunto em português era, então, praticamente nula, mas, mesmo assim, quando encontrada, limitava-se ao básico do básico.

No segundo semestre de 2003, estimulado pela completa falta de material de consulta gratuita na internet e impulsionado por sua veia de educador desenvolvida durante anos em sala de aula, decidiu lançar o site CSS para Web Design, o qual é nacionalmente conhecido como o site Maujor, hospedado em `http://www.maujor.com/`. A proposta inicial do site era a de divulgar a cascading style sheet (CSS), ou folha de estilo em cascata, com base no compartilhamento de suas experiências com tal técnica. Com a pronta aceitação e o sucesso crescente do site, o objetivo inicial tornou-se mais ambicioso e passou a ser a divulgação das Web Standards.

No início de 2006, com a popularização e a adesão maciça aos blogs, criou o Blog do Maujor hospedado em `http://www.maujor.com/blog/`, com propósito semelhante ao do site, mas com uma dinâmica mais ativa e a efetiva participação de seus leitores.

Tanto o site como o blog constituem-se em referência nacional para Web Standards e, sem dúvida, foram e continuam sendo um dos grandes responsáveis pela divulgação e popularização das Web Standards, assim como, em particular, das CSS no Brasil.

Maujor, como é conhecido o autor, é ainda um ativo frequentador de fóruns, escreve para vários portais brasileiros voltados a desenvolvedores web, é autor de artigos em inglês e de trabalhos relacionados às CSS publicados em sites internacionais sobre Web Standards. Ocupa o segundo lugar na lista de tradutores mundiais de documentos do W3C por quantidade de documentos versados. Publicou em seu site mais de 40 traduções de artigos sobre Web Standards escritos por consagrados autores internacionais. Traduziu para o português as ferramentas para testes de acessibilidade em sites, denominadas Analisador de contraste de cores e Barra de ferramentas para acessibilidade na web, ambas em parceria com o WAT-C, um consórcio australiano voltado ao desenvolvimento de ferramentas destinadas a testes de acessibilidade.

Introdução

Este livro tem por objetivo fornecer aos profissionais envolvidos com o desenvolvimento para a web os conceitos fundamentais e técnicas de programação necessárias ao emprego da biblioteca jQuery na criação de efeitos especiais e obtenção de comportamentos de variadas naturezas em sites web, tornando-os mais interativos e visualmente mais agradáveis e amigáveis. Aborda o uso da linguagem de programação JavaScript segundo a sintaxe especificamente criada para fazer funcionar e tornar usável a biblioteca jQuery.

O livro está dividido em duas partes como descrito a seguir.

Primeira parte – Fundamentos teóricos da biblioteca jQuery

A primeira parte compreende os capítulos de 1 a 7 e tem por objetivo apresentar a biblioteca, examinar sua sintaxe geral e estudar o emprego e finalidade dos métodos, propriedades e diferentes funcionalidades disponíveis na biblioteca JQuery.

No estudo dos métodos, adotou-se um modelo que consiste na apresentação da sintaxe do método seguida de uma explicação da sua finalidade, em um pequeno trecho de código de emprego real demonstrando o uso geral do método e, finalmente, na disponibilização de um arquivo HTML para download ou consulta on-line, hospedado no site do livro, contendo a demonstração prática do método estudado.

Segunda parte – jQuery na prática

A segunda parte do livro compreende os capítulos de 8 a 14 e tem por objetivo demonstrar efeitos jQuery de variadas naturezas e de emprego real em um site.

Cada capítulo desta parte está estruturado para estudar efeitos em elementos específicos do HTML. O capítulo 8 introduz a parte prática do livro e dedica-se ao estudo dos efeitos de animação básicos com ênfase nas técnicas de revelar e esconder conteúdos marcados com diferentes elementos HTML.

O capítulo 9 mostra a criação de efeitos para esconder e revelar conteúdos, demonstrando as técnicas aplicadas a uma página contendo um FAQ CSS e a outra, contendo notícias.

O capítulo 10 dedica-se aos efeitos em tabelas de apresentação de dados, estudando técnicas para destacar trechos delas que recebem o foco do usuário, bem como maneiras de apresentar tabelas extensas, com mecanismos de revelação de trechos destas.

O capítulo 11 aborda os formulários HTML, mostrando os efeitos fundamentais normalmente implementados no desenvolvimento deles, tais como criação de máscaras e dicas de preenchimento de campos.

No capítulo 12, examinam-se os efeitos em imagens, abordando não somente a manipulação de imagens individualmente, mas também a criação de galerias de imagens.

O capítulo 13 dedica-se à criação de efeitos em mecanismos de navegação.

No capítulo 14 examinamos o caso real de um menu tipo sanfona. Trata-se do menu do site `www.maujor.com`. Ensinamos não só o script jQuery que faz funcionar o menu, mas também as técnicas CSS da sua construção.

Para quem foi escrito este livro

Este livro foi escrito para aquelas pessoas envolvidas na criação de sites tanto na área de design quanto na de desenvolvimento e programação, sem conhecimento das técnicas jQuery ou com conhecimentos superficiais.

O objetivo do livro é fornecer informações detalhadas do funcionamento básico da biblioteca, estudando seus métodos e funcionalidades. Explicações teóricas em linguagem corrente e clara, dispensando, sempre que possível, o jargão técnico avançado, são acompanhadas de exemplos práticos explicados passo a passo e complementados por arquivo HTML para consulta. Não se abordaram técnicas avançadas de emprego da biblioteca, construção de plug-ins, interação com AJAX nem outra linguagem ou biblioteca.

Para tirar o máximo proveito dos ensinamentos contidos em cada capítulo é pré-requisito conhecer os fundamentos e a sintaxe da linguagem JavaScript, não sendo necessário um conhecimento profundo dela, bem como razoável conhecimento de folhas de estilo, em particular seletores CSS.

Os conceitos e o entendimento das técnicas de desenvolvimento com jQuery são uma poderosa ferramenta de apoio na criação de sites mais interativos e agradáveis, enriquecendo a experiência do usuário. Profissionais da área de design, familiarizados com as técnicas aqui descritas, contarão com uma valiosa fonte de inspiração no planejamento das funcionalidades para incrementar suas criações.

Os iniciantes, de posse das noções básicas da linguagem JavaScript, irão se beneficiar deste livro por iniciar seus estudos em uma fonte que aborda as mais modernas técnicas de escrita do código, ensejando uma mudança no rumo de seu estudo que irá reduzir a curva de aprendizado e acelerar tal processo. Não se intimidem com conceitos ou terminologias que lhes sejam completamente desconhecidos nos primeiros capítulos. Com a sequência de leitura, as dúvidas tenderão a desaparecer naturalmente.

Convenções tipográficas

Com a finalidade de destacar diferentes tipos de informação neste livro, adotaram-se algumas convenções tipográficas mostradas a seguir.

Dica

Texto contendo uma dica sobre o assunto tratado:

O valor do atributo `src` indica o caminho (diretório) no qual se encontra gravado o arquivo da biblioteca.

Alerta

Texto contendo um lembrete sobre procedimento extra com relação ao assunto tratado:

jQuery, ao contrário de JavaScript, não requer loops para construir arrays quando busca elementos no DOM. O construtor `$()` armazena tudo que encontra, automaticamente, em um objeto array.

Terminologia

Texto estabelecendo a adoção de grafia-padrão em todo o livro para termos ou frases com mais de uma terminologia, tradução ou significado:

Nos códigos desenvolvidos neste livro, adotou-se a sintaxe: $() .

Chamada

Uma chamada para uma seção anterior na qual o assunto em questão foi explicado com detalhes.

Por exemplo:

Esta função destina-se a servir de container para scripts jQuery que devam ser executados somente após o carregamento do DOM. É uma função que substitui o método `ready()` estudado em [C1 S1.1.6.2]. Neste exemplo a chamada é para a seção 1.1.6.2 do capítulo 1.

Marcação e scripts

Nas marcações e scripts que exemplificam a teoria, transcreveram-se somente os trechos que interessam ao assunto tratado. Omitiram-se os trechos que não dizem respeito ou não são relevantes para o entendimento do assunto, para não ocupar espaço desnecessário no livro.

Blocos de marcação são marcados com fonte monoespaçada:

```
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('#cor').css('color','#FF0000');
      });
   });
</script>
```

Trechos de marcação que merecem destaque são marcados em negrito:

```
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
```

```
        $('#btn-vermelha').click(function() {
            $('#cor').css('color','#FF0000');
        });
    });
</script>
```

Para explicar passo a passo cada linha de um script, este é apresentado com suas linhas numeradas e, a seguir, os comentários são referenciados ao número da linha comentada:

```
1.      ...
2.      $('<h4 class="legenda"></h4>').insertAfter('legend');
3.          var textoLegenda = $('legend').remove().text();
4.          $('.legenda').append(textoLegenda);
5.          $('fieldset').addClass('fieldset');
6.      });
```

Código comentado:

Linha	Descrição
Linha 2	Insere logo após o elemento `legend` um elemento `h4` com a classe `legenda` e vazio.
Linha 3	Armazena o texto do elemento `legend` em uma variável denominada `textoLegenda` e remove o elemento do DOM.
Linha 4	Insere dentro do elemento `h4` criado anteriormente o texto da legenda.
Linha 5	Define a classe `fieldset` para o elemento `fieldset`, para fins de estilização.

Códigos, marcações e sintaxe CSS contidos em textos são marcados com fonte monoespaçada.

A função `jQuery.noConflict()` permite, entre outras funcionalidades, criar um pseudônimo personalizado para desenvolvimento.

Arquivos para download

Os scripts mostrados no livro estão disponíveis para download e/ou consulta on-line no site do livro, em `http://www.livrojquery.com.br`. Os documentos estão separados em pastas por capítulos e foram nomeados com sintaxe própria, conforme o exemplo a seguir:

```
arquivo-2.1d.html
```

Este é um arquivo que mostra um script contido no primeiro item do segundo capítulo (2.1) e é o quarto script desse item (letra d no final do nome do arquivo).

Adicionalmente, ao final do script, há uma indicação do arquivo no qual foi inserido conforme convenção mostrada no exemplo a seguir:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(function($) {
      $('div, p').css('background', 'green');
   });
</script>
</head>
<body>
<div>DIV</div>
<p>Parágrafo</p>
<div>DIV</div>
</body>
</html>
```

 [arquivo-2.1d.html] ⇦ Indicação do arquivo que demonstra o funcionamento do script no site do livro. Trata-se do quarto arquivo (letra d) do item 2.1 no segundo capítulo.

Destaques em geral

Palavras ou termos cujo significado deva ser destacado são grafados em *itálico*.

Por exemplo:

jQuery destina-se a adicionar interatividade e dinamismo *incrementando de forma progressiva* e *não obstrutiva* a usabilidade, a acessibilidade e o design.

Variáveis

Valores variáveis em códigos são grafados em *itálico*.

Por exemplo:

```
jQuery(expressão, [contexto])
```

Site do livro

O site de suporte a este livro está localizado em `http://www.livrojquery.com.br`.

No site, incluíram-se as facilidades relacionadas a seguir:

- **Arquivo de códigos**: os códigos dos exemplos mostrados no livro estão disponíveis tanto para consulta on-line como para download.
- **Errata**: efetuou-se um exaustivo trabalho de revisão tipográfica. Contudo, a prática mostra que nenhum livro está isento de erros – e com este não há de ser diferente. Uma página do site dedicada à errata está disponível para consulta.
- **Feedback**: incentiva-se vivamente os leitores a emitir opinião sobre qualquer aspecto do livro. Serão de grande valia informações sobre qualquer erro detectado para aperfeiçoar futuras edições e atualizar a errata. Você pode se comunicar com a editora pelo e-mail `novatec@novatec.com.br` ou diretamente com o autor pelo e-mail `maujorcss@maujor.com`.

Parte I

Fundamentos teóricos da biblioteca jQuery

A primeira parte deste livro compreende os capítulos de 1 a 7 e tem por objetivo apresentar a biblioteca, examinar sua sintaxe geral e estudar o emprego e finalidade dos métodos, propriedades e diferentes funcionalidades disponíveis na biblioteca JQuery.

CAPÍTULO 1

O que é jQuery?

Neste capítulo, será apresentada a biblioteca jQuery, relatando-se suas origens, finalidades e destinação. Será feito um breve relato histórico de sua evolução, examinando as motivações que resultaram em sua criação, e serão descritas algumas de suas possibilidades, dando uma ideia do seu potencial e mostrando a força e poder da programação JavaScript com o uso da biblioteca jQuery.

1.1 Definições e conceitos

1.1.1 O que é jQuery?

jQuery é uma biblioteca JavaScript criada por John Resig e disponibilizada como software livre e aberto, ou seja, de emprego e uso regido segundo licença conforme as regras estabelecidas pelo MIT (Massachusetts Institute of Technology) e pelo GPL (GNU General Public License). Isto, resumidamente, significa que você pode usar a biblioteca gratuitamente tanto para desenvolver projetos pessoais como comerciais. Para maiores detalhes sobre esses tipos de licença, consulte os seguintes endereços na internet:

- `http://pt.wikipedia.org/wiki/Mit_license`
- `http://pt.wikipedia.org/wiki/GNU_General_Public_License`

John Resig, no prefácio do livro *jQuery in Action*, diz o seguinte:

> "O foco principal da biblioteca jQuery é a simplicidade. Por que submeter os desenvolvedores ao martírio de escrever longos e complexos códigos para criar simples efeitos?".

Sem dúvida, ele estava em um momento de rara inspiração quando assim escreveu, pois soube resumir muito bem jQuery. É uma maneira simples e fácil de escrever JavaScript colocada ao alcance não só de programadores experientes, mas também de designers e desenvolvedores com pouco conhecimento de programação.

E o que torna o estudo e entendimento dos conceitos básicos de jQuery mais fascinante ainda é o fato de que você não precisa ser um profundo conhecedor de JavaScript, por mais estranho que isso possa parecer, pois se trata de uma biblioteca criada para ser usada com base nessa programação.

Simplicidade é a palavra-chave que resume e norteia o desenvolvimento com jQuery. Linhas e mais linhas de programação JavaScript escritas para obter um simples efeito em um objeto são substituídas por apenas algumas, escritas com sintaxe jQuery. Intrincados e às vezes confusos códigos JavaScript destinados a selecionar um determinado elemento HTML componente da árvore do documento são substituídos por um simples método jQuery. Você mesmo, ao longo da leitura deste livro, irá constatar como jQuery consegue a proeza de criar extraordinários efeitos com uma simplicidade impressionante, colocando o desenvolvimento de scripts a seu alcance e dispor imediatos, sem exigir profundos conhecimentos de programação.

1.1.1.1 Histórico

No dia 22 de agosto de 2005, John Resig, cuja foto é mostrada na figura 1.1, um desenvolvedor americano profundo conhecedor de JavaScript, que atua na Corporação Mozilla e é autor do livro *Pro JavaScript Techniques*, escreveu em seu blog um artigo relatando sua frustração com a maneira verbosa de se escrever JavaScript para obter os resultados pretendidos.

Figura 1.1 – John Resig, o criador da jQuery.

Nesse artigo, publicou alguns exemplos no quais propunha o uso de seletores CSS com o objetivo de simplificar e dar maior versatilidade ao código. Escreveu, então, que essa, ainda, não era a forma definitiva do que tinha em mente, mas iria aperfeiçoar e testar suas propostas. O nome ainda não existia, mas nessa ocasião foi lançada a ideia que traria como resultado a biblioteca jQuery.

Aproximadamente cinco meses após a publicação do artigo em seu blog, John Resig apresentou publicamente os resultados de seus estudos em uma palestra intitulada "jQuery, a nova onda para JavaScript", proferida no BarCampNYC – Wrap Up, no dia 14 de janeiro de 2006.

O ano de 2006 marcou a criação do primeiro plug-in para a biblioteca, o lançamento de uma versão não obstrutiva de LightBox usando a biblioteca jQuery. Nesse ano, ocorreram, ainda, o lançamento das versões 1.0, 1.0.1, 1.0.2, 1.0.3 e 1.04, da versão XML da biblioteca e o primeiro conteste público de criação com jQuery.

Em 2007, lançaram-se as versões 1.1, 1.1.1, 1.1.2, 1.1.3a, 1.1.3, 1.1.3.1, 1.1.4, 1.2 e 1.2.1. No dia 11 de março ocorreu o primeiro encontro jQuery.

Em 2008, até a data em que escrevi este livro, foram lançadas as versões 1.2.2, 1.2.3, 1.2.4, 1.2.5 e 1.2.6.

1.1.2 Para que serve jQuery?

jQuery destina-se a adicionar interatividade e dinamismo às páginas web, proporcionando ao desenvolvedor funcionalidades necessárias à criação de scripts que visem a incrementar, de forma progressiva e não obstrutiva, a usabilidade, a acessibilidade e o design, enriquecendo a experiência do usuário.

Use jQuery em sua página para:

- adicionar efeitos visuais e animações;
- acessar e manipular o DOM;
- buscar informações no servidor sem necessidade de recarregar a página (fora do escopo deste livro);
- prover interatividade;
- alterar conteúdos;
- modificar apresentação e estilização;

- simplificar tarefas específicas de JavaScript;
- realizar outras tarefas relacionadas às descritas.

1.1.3 jQuery em conformidade com os Padrões Web

jQuery foi criada com a preocupação de ser uma biblioteca em conformidade com os Padrões Web, ou seja, compatível com qualquer sistema operacional e navegador, além de oferecer suporte total para as CSS 3. Sim, é isso mesmo, você pode usar todos os poderosos seletores previstos nas CSS 3 sem se preocupar se este ou aquele navegador suporta o seletor para as CSS. Esclarecendo: a sintaxe do seletor segue a sintaxe prevista nas CSS 3, mas quem usa os seletores é a biblioteca, com a finalidade de selecionar elementos no DOM e não estilizar.

Evidentemente, isso não significa que todo código escrito com uso de jQuery resulta em um documento válido segundo os Padrões Web. A biblioteca foi criada e está de acordo com as diretrizes do W3C, mas cabe a você, desenvolvedor, escrever seus scripts em conformidade.

Se você pretende escrever em conformidade com os Padrões Web, adote como regra básica de desenvolvimento de scripts a filosofia de que jQuery se destina a adicionar interatividade e dinamismo, *incrementando de forma progressiva* e *não obstrutiva* a usabilidade, a acessibilidade e o design com o propósito de enriquecer a experiência do usuário.

Leia a analogia a seguir.

Por não dispor de verba suficiente, Fulano compra um carro, o modelo mais simples da linha, e alguns meses depois, tendo sobrado uma verba, resolve investir no carro equipando-o com alguns acessórios, como insulfilm, som, alarme, protetor solar e alguns outros itens, mudando o aspecto inicial do carro e fazendo-o parecer um modelo intermediário da linha. Passado algum tempo e tendo economizado uma boa quantia, Fulano resolve melhorar o carro instalando calotas de aço escovado, GPS, TV digital, frigobar, MP3 e outros acessórios mais sofisticados, conseguindo um visual de carro top de linha.

Fulano é o desenvolvedor, o carro mais simples é a página web sem scripts, o carro top de linha é a página web incrementada com jQuery, as duas etapas de colocação de acessórios representam o incremento progressivo e o fato de Fulano (e ninguém em sã consciência) não ter mandado retirar o motor do carro para colocar um jarro de flores representa o incremento não obstrutivo.

O carro top de linha de Fulano continuará sendo um carro com todas as suas funcionalidades, cumprindo rigorosamente todas para as quais foi projetado quando saiu da fábrica, mesmo que dele sejam retirados todos os acessórios. Com scripts em geral, e jQuery no nosso caso, acontece a mesma situação. Seu documento estará em conformidade com os Padrões Web se continuar usável e funcional caso os scripts parem de funcionar. Um belíssimo menu de navegação, com efeitos ultrassofisticados, não valerá nada se não for acessível sem o script que o faz funcionar. Pode tornar-se esteticamente horroroso, mas deve cumprir sua finalidade quando não sustentado pelo script.

Algumas técnicas para preservar a acessibilidade aos conteúdos incrementados com jQuery são básicas e serão abordadas em momento oportuno. Outras situações de preservação de acessibilidade por estarem inseridas no contexto de um desenvolvimento particular não têm solução em uma técnica preestabelecida e dependem da criatividade e capacidade de o desenvolvedor resolver situações específicas.

Sempre que for o caso, os exemplos deste livro serão desenvolvidos de forma não obstrutiva, contudo, como dito anteriormente, quando a obstrução depender de um contexto particular, será entendido que sua avaliação e solução dependem de cada caso.

1.1.4 Características da biblioteca jQuery

jQuery é uma biblioteca JavaScript que possui as seguintes características:

- utiliza seletores CSS para localizar elementos componentes da estrutura de marcação HTML da página;
- possui arquitetura compatível com instalação de plug-ins e extensões em geral;
- é indiferente às inconsistências de renderização entre navegadores;
- é capaz de interação implícita, isto é, não há necessidade de construção de loops para localização de elementos no documento;
- admite programação encadeada, ou seja, cada método retorna um objeto.
- é extensível, pois admite criação e inserção de novas funcionalidades na biblioteca existente.

Não se preocupe se algumas das características descritas soem sem sentido para você. Com o prosseguimento da leitura, os conceitos ficarão claros e compreensíveis. O fato é que tais características possibilitaram a criação de uma biblioteca bastante compacta e, ao mesmo tempo, poderosa o bastante para oferecer funcionalidades que tornam o processo de desenvolvimento igualmente compacto e extremamente simples.

Por ser distribuída como software livre, jQuery tem o apoio e o envolvimento de uma considerável comunidade. Desenvolvedores do mundo todo têm contribuído em larga escala com novas ideias, scripts, plug-ins, extensões e toda sorte de implementações, com a finalidade de incrementar não só a biblioteca, mas também as técnicas de desenvolvimento jQuery.

1.1.5 Como instalar jQuery

A biblioteca jQuery nada mais é que um arquivo JavaScript (arquivo do tipo texto puro gravado com a extensão `.js`, como `meu_arquivo.js`) que deverá ser linkado à página web onde serão aplicados efeitos, ou seja, a página web onde serão aplicados os efeitos deverá "chamar" biblioteca. É somente isso. Você não precisará instalar nada. Basta fazer o download gratuito do arquivo e "chamá-lo" na(s) página(s).

Uma página ou documento "chama" um arquivo de script fazendo uso do elemento `script` e seus atributos `type` e `src` colocados na seção `head` do documento.

A versão mais recente da biblioteca está no arquivo `jquery-1.2.6.js` e adiante você verá como e onde fazer o download dele. É muito provável que na ocasião em que você estiver lendo este tópico já exista uma versão mais recente e é esta que você deverá baixar para seus estudos e desenvolvimentos. Então, um documento que use a biblioteca deverá ter marcado, em sua seção head, o seguinte elemento script, entre outros elementos próprios de cada caso:

```
...
<head>
...
<script type="text/javascript" src="/caminho/jquery-1.2.6.js"></script> <!-- esta
   linha chama a biblioteca jQuery -->
</head>
...
```

O valor do atributo `src` indica o caminho (diretório) no qual se encontra gravado o arquivo da biblioteca.

A partir daqui é imperativo ter gravado em seu HD a última versão da biblioteca para poder acompanhar os exercícios deste livro e fazer funcionar seus experimentos com jQuery, que, tenho certeza, irão fasciná-lo.

Entre no site oficial `http://jquery.com`, vá para a página de download conforme mostrado na figura 1.2 e faça o download da última versão da biblioteca jQuery.

Figura 1.2 – Site oficial da biblioteca jQuery.

Observe, na área de download em destaque na figura 1.2, que existem diferentes tipos de apresentação da biblioteca para download e um link para um documento hospedado no próprio site, contendo um relatório das modificações introduzidas na atual versão.

A apresentação denominada "Uncompressed" – `jquery-1.2.6.js` – destina-se ao uso em testes, estudos e desenvolvimento da biblioteca. Trata-se de um arquivo no qual o texto do script foi escrito com espaço entre cada linha de código, amplamente comentado, o que facilita a leitura, análise e entendimento do código contido no arquivo. Esta é a apresentação recomendada para você baixar e fazer uso nos estudos e acompanhamento dos experimentos desenvolvidos neste livro. O tamanho desse arquivo é de 97,8KB.

A apresentação denominada "Minified" – `jquery-1.2.6.min.js` – difere da anterior por terem sido removidos todos os comentários e espaçamentos entre linhas e declarações, tornando o código difícil de ler para humanos e mais compacto, pois se transforma em uma sequência corrida de código, sem quebras de linha. Esta é a apresentação recomendada para ser usada no desenvolvimento de sites. A vantagem sobre a versão anterior é que se trata do mesmo arquivo com tamanho reduzido para 54.4KB.

Finalmente, a apresentação denominada "Packed" – `jquery-1.2.6.pack.js` – é uma versão obtida por processo de compressão JavaScript da biblioteca, resultando em um arquivo com tamanho igual a 30.3KB. Essa versão, por ser codificada, não é legível para humanos.

Embora com tamano menor que o da versão "Minified" o tempo de carregamento desta versão acaba sendo praticamente igual ao daquela versão, pois há que se computar o tempo de descompressão quando o usuário recebe a página. Esta versão tem a desvantagem em relação à anterior de não ir para o cache, tendo que ser carregada novamente toda vez que o usuário volta ao site. Outra desvantagem que desaconselha seu uso é o fato de que o processo de descompressão pode, eventualmente, ser imperfeito, introduzindo bugs não existentes na versão sem compressão ou na compacta. A não ser que você tenha uma boa justificativa para usá-la, não use-a.

A partir daqui, para testar os exemplos do livro, supõe-se que você baixou e gravou em seu HD a apresentação "Uncompressed" – `jquery-1.2.6.js` – da biblioteca, para ser chamada pelas páginas de estudos conforme explicado anteriormente.

1.1.6 jQuery na prática

Para que um script consiga imprimir dinamismo, alterar comportamentos ou apresentação, no todo ou em partes de uma página web, precisa de um método de acesso à árvore do documento com a finalidade de encontrar o elemento HTML no qual será vinculado o comportamento.

Para exemplificar, imagine uma página web na qual existe um botão que, ao ser clicado, muda a cor de um cabeçalho, de verde para vermelha.

Nos exemplos constantes dos arquivos disponíveis para consulta ou download, você irá verificar o funcionamento dos scripts, interagindo com a página que contém o exemplo, clicando um botão ou agindo com o mouse. Para repetir o funcionamento do script, recarregue a página (em ambiente Windows tecle F5).

▶ **Solução com JavaScript in-line**:

```
...
<head>
...
<style type="text/css" media="all">
   h1 {color:#090;} /* cor verde para o cabeçalho */
</style>
</head>
<body>
   <h1 id="cor">Cabeçalho muda de cor</h1>
   <button type="button" onclick = "document.getElementById('cor').style.color='#FF0000';">
      Vermelha
   </button>
...
```

 [arquivo-1.1.6a.html]

A técnica usada nessa solução consiste em inserir script dentro de marcação HTML (in-line), sendo uma prática ultrapassada e, infelizmente, ainda amplamente empregada por muitos desenvolvedores. Essa solução contraria um princípio fundamental dos Padrões Web que preconiza separação total entre estrutura de marcação com HTML, apresentação com CSS e comportamento com scripts. Cada código no seu arquivo correspondente e separado. Evite usar essa solução.

▶ **Solução com função JavaScript**:

```
...
<head>
...
<style type="text/css" media="all">
   h1 {color:#090;}
</style>
<script type="text/javascript">
   function mudaCor() {
      document.getElementById('cor').style.color = '#FF0000';
   }
</script>
</head>
<body>
   <h1 id="cor">Cabeçalho muda de cor</h1>
   <button type="button" onclick = "mudaCor();">
      Vermelha
   </button>
...
```

 [arquivo-1.1.6b.html]

A técnica usada nessa solução é uma melhoria da solução anterior e consiste em definir uma função dentro da seção `head` do documento, que pode ser chamada por vários botões dentro de um mesmo documento, permitindo ampliar o uso do script para além de um botão. Uma variante dessa solução consiste em colocar a função em um arquivo externo e linkar o arquivo ao documento. Isso permite usar a função em vários botões dentro de vários documentos. Essa solução continua misturando estrutura com comportamento e, tal como a anterior, deve ser evitada.

▶ **Solução com JavaScript fora da marcação**:

```
...
<head>
...
<style type="text/css" media="all">
   h1 {color:#090;}
</style>
<script type="text/javascript">
   window.onload = function() {
      document.getElementById('btn-vermelha').onclick = mudaCor;
   };

   function mudaCor() {
      document.getElementById('cor').style.color = '#FF0000';
   };
</script>
</head>
<body>
   <h1 id="cor">Cabeçalho muda de cor</h1>
   <button type="button" id="btn-vermelha">Vermelha</button>
...
```

[arquivo-1.1.6c.html]

Agora, sim! A marcação HTML está isenta de código JavaScript, pois se incorporou o script à seção `head` do documento. Resta agora colocar o script em um arquivo externo separado e linkar para a(s) página(s). Essa é uma boa solução sob o ponto de vista da separação do comportamento, marcação e estilização.

▶ **Solução com jQuery**:

Caso você nada conheça de jQuery, não se intimide com o script a seguir. Limite-se a observar com atenção cada linha do código que seu conhecimento de JavaScript lhe dará uma noção bem próxima de seu funcionamento. Com o prosseguimento da leitura, você entenderá a finalidade de cada linha do script:

```
...
<head>
...
<style type="text/css" media="all">
   h1 {color:#090;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('#cor').css('color","#FF0000');
      });
   });
</script>
</head>
<body>
   <h1 id="cor">Cabeçalho muda de cor</h1>
   <button type="button" id="btn-vermelha">Vermelha</button>
...
```

[arquivo-1.1.6d.html]

Ao contrário das soluções tradicionais com JavaScript, como as mostradas anteriormente, o uso da biblioteca jQuery não permite misturar script com marcação HTML. Não é previsto nem existe uma sintaxe para jQuery in-line. Você é obrigado a inserir seu script incorporado na seção `head` do documento ou escrevê-lo em um arquivo separado e linkar para os documentos onde serão utilizados. Como regra geral, adote as mesmas diretrizes que regem a vinculação de folhas de estilo, isto é, se seu script serve a mais de uma página, adote a solução de linkar, senão, a solução de incorporar na seção `head` do documento. Mas lembre-se que mesmo para uso em uma só página, em scripts que demandem tempo de carregamento não desprezível, a solução de linkar é melhor, pois neste caso o script vai para o cache da máquina do usuário.

Os quatro exemplos aqui apresentados tiveram por objetivo único mostrar na prática um efeito bem simples criado de três maneiras diferentes com JavaScript

e à maneira jQuery. Ainda que seus conhecimentos básicos de JavaScript sejam limitados, não se intimide, simplesmente consulte com atenção os códigos e vá em frente. Mas tanto você quanto aqueles familiarizados com JavaScript não deixem de abrir cada uma das páginas contendo os scripts, constatar seu funcionamento e olhar no código-fonte da página. Os arquivos das páginas estão no site do livro, disponíveis para consulta on-line e também para download.

Toda página na qual se pretende fazer funcionar um script jQuery deverá estar linkada para o arquivo da biblioteca baixado do site jQuery.

1.1.6.1 Evento `window.onload`

JavaScript é uma técnica de programação que funciona percorrendo e buscando seus alvos (elementos da marcação) na árvore do documento ou no DOM, ou seja, um script só consegue executar sua ação se todo o documento já tiver sido carregado. Os elementos da marcação de uma página só existem depois que a página é carregada, ou, mais precisamente, vão existindo à medida que a página vai sendo carregada e na sequência em que aparecem na marcação a partir da declaração do `DOCTYPE` até a tag de fechamento do elemento `html`.

Na prática, isso significa que se você inserir na marcação de uma página um script que se refira a um elemento `h1`, por exemplo, em local acima daquele no qual foi escrito o elemento `h1`, seu script não irá funcionar, porque será carregado na página antes do carregamento do elemento `h1`.

Observe, a seguir, a transcrição do terceiro exemplo mostrado no item 1.1.6 no qual todo JavaScript foi retirado da marcação e passado para a seção `head` do documento.

```
...
<head>
...
<style type="text/css" media="all">
   h1 {color:#090;}
</style>
<script type="text/javascript">
   window.onload = function() {
      document.getElementById('btn-vermelha').onclick = mudaCor;
   };
   function mudaCor() {
      document.getElementById('cor').style.color = '#FF0000';
   };
```

```
</script>
</head>
<body>
   <h1 id="cor">Cabeçalho muda de cor</h1>
   <button type="button" id="btn-vermelha">Vermelha</button>
...
```

Vamos alterar ligeiramente o script anterior reescrevendo-o conforme mostrado a seguir.

```
...
<head>
...
<style type="text/css" media="all">
   h1 {color:#090;}
</style>
<script type="text/javascript">
   document.getElementById('btn-vermelha').onclick = mudaCor;
   function mudaCor() {
      document.getElementById('cor').style.color = '#FF0000';
   }
</script>
</head>
<body>
<h1 id="cor">Cabeçalho muda de cor</h1>
<button type="button" id="btn-vermelha">Vermelha</button>
...
```

[arquivo-1.1.6.1a.html]

Observe que o que se fez foi retirar do script a declaração que diz:

```
window.onload = function() {
   ...faça isso...
}
```

Mesmo para quem pouco conhece de JavaScript, a declaração é autoexplicativa, pois `window.onload`, traduzindo para o português, significa: depois que o documento for carregado, faça isso. Faça isso é a função `mudaCor()`.

Agora, com a retirada que se fez, o script (`...faça isso...`) não esperará a página ser carregada e, consequentemente, não funcionará e o botão não trocará a cor do cabeçalho, conforme se pode constatar no arquivo existente no site do livro. Isso ocorre pela razão explicada anteriormente: o script foi escrito antes dos elementos `h1` e `button` com seus ids `cor` e `btn-vermelha` constantes do script, ou seja, quando o script é carregado, ainda não existem os ids de que precisa para funcionar.

Se tirar o script da seção `head` do documento e posicioná-lo depois dos elementos envolvidos, funcionará normalmente. Faça assim:

```
...
<head>
...
<style type="text/css" media="all">
   h1 {color:#090;}
</style>
</head>
<body>
   <h1 id="cor">Cabeçalho muda de cor</h1>
   <button type="button" id="btn-vermelha">Vermelha</button>
<script type="text/javascript">
   document.getElementById('btn-vermelha').onclick = mudaCor;
   function mudaCor() {
      document.getElementById('cor').style.color = '#FF0000';
   }
</script>
...
```

[arquivo-1.1.6.1b.html]

E tudo voltará a funcionar. Veja nos arquivos existentes no site do livro as páginas mostrando as duas situações descritas anteriormente.

Conclusão

Para possibilitar a retirada de seus scripts da marcação HTML, a linguagem JavaScript dispõe da declaração `window.onload` que atua como uma espécie de aviso para que a função entre em ação (ou comece a "rodar") somente após a página ter sido completamente carregada.

Existem outros métodos que cumprem a mesma finalidade e oferecem outras funcionalidades, como permitir declarar várias funções, mas não é o escopo deste livro aprofundar a linguagem JavaScript.

1.1.6.2 Método `ready()`

Tal como vimos para JavaScript, jQuery, que se baseia nessa linguagem, também precisa que seus scripts conheçam a árvore do documento para poder funcionar.

Na sintaxe jQuery, o equivalente ao `window.onload` e todas as suas variantes é mostrado a seguir.

```
$(document).ready(function() {
    ...faça isso...
});
```

Essa notação é conhecida como *sintaxe formal* para escrita do método `ready()`, que significa o seguinte: "quando o documento estiver pronto, faça isso". Faça isso é o script a executar.

Você pode omitir o parâmetro `document` passado para a função jQuery construtora `$()` e escrever com a seguinte sintaxe:

```
$().ready(function() {
    ...faça isso...
});
```

O método jQuery `read()` oferece duas vantagens sobre seu equivalente JavaScript:

- O script está pronto para funcionar tão logo a árvore do documento tenha sido carregada. Não é necessário que todos os componentes da página, tais como imagens, folhas de estilo, animações e mídias em geral, tenham sido carregados. Basta que a estrutura de marcação da página esteja disponível e o script já poderá funcionar.
- Não há variações funcionais para o método e pelo fato de jQuery não admitir mistura de script com marcação, utiliza-se a sintaxe mostrada para servir de container aos scripts.

Existe uma sintaxe alternativa à sintaxe formal mostrada anteriormente para o método `ready()`, chamada de *sintaxe abreviada*, que é a seguinte:

```
$(function() {
    ...faça isso...
});
```

Nos códigos desenvolvidos neste livro, será adotada a sintaxe formal, pois se considera que, apesar de ser mais extensa, oferece melhor legibilidade, sendo em consequência mais fácil de ser visualizada. Em seus desenvolvimentos reais, sinta-se à vontade para usar a sintaxe abreviada que, certamente, reduz o trabalho de digitação.

1.1.7 Fundamentos jQuery

A finalidade do uso de jQuery é controlar o comportamento de toda ou partes de uma página web. Sabe-se que uma página web nada mais é do que marcação HTML. Então, é lícito concluir que o princípio de funcionamento de jQuery fundamenta-se em sua capacidade de encontrar os elementos HTML que constituem a página e a estes anexar seus métodos.

1.1.7.1 Construtor jQuery

Inicialmente, veja o encarregado de encontrar elementos HTML em um documento:

```
$()
```

Tecnicamente, trata-se do que se denomina, em linguagem de programação, de *construtor*. O construtor `$()` ou, ainda, a função construtora `$()` estará presente em todos os scripts que você escrever, portanto decore desde já sua sintaxe: um sinal de cifrão seguido de parênteses (o que, convenha, não é tão complicado de decorar).

Outras bibliotecas JavaScript também usam a função `$()`. Isso pode causar conflitos e mau funcionamento de scripts quando se usa mais de uma biblioteca no mesmo documento. Nesses casos, existem métodos para evitar conflitos e um deles é usar a sintaxe alternativa mostrada a seguir. Outros métodos para evitar conflitos serão apresentados posteriormente.

```
jQuery()
```

Simplicidade é a filosofia que orienta o desenvolvimento de jQuery desde sua criação. Observe os códigos a seguir. Você, mesmo não conhecendo nada de jQuery, seria capaz de concluir a finalidade de cada uma das linhas do código a seguir?

```
$('h1');
$('p');
$('span');
$('table');
```

Bem simples, não é mesmo? Você está instruindo seu construtor a encontrar *todos* os elementos `h1`, `p`, `span` e `table` respectivamente.

jQuery, ao contrário de JavaScript, não requer loops para construir arrays quando busca elementos no documento. O construtor `$()` armazena automaticamente em um objeto array tudo que encontra.

Encontrar *todos* os elementos de um determinado tipo não parece muito útil. Seria bem melhor se você pudesse encontrar uma ocorrência específica de um elemento.

Por exemplo: quero encontrar o terceiro parágrafo do documento. Bem, neste caso, uma solução seria marcar o terceiro parágrafo com o atributo `id` para identificar sua ocorrência:

```
<p id="xpto">Texto do terceiro parágrafo</p>
```

e escrever o construtor assim:

```
$('#xpto')
```

Você conhece CSS? Ainda não? É essencial começar a aprender hoje mesmo, sob pena de ficar ultrapassado nas técnicas de desenvolvimento web. jQuery não é exceção à regra e faz amplo emprego de seletores CSS. Como se disse anteriormente, adota os poderosos seletores CSS 3 para localizar elementos no documento. E não se preocupe, pois o uso de seletores CSS em jQuery em nada se relaciona com suporte pelos navegadores. Use e abuse desses seletores que seus scripts funcionarão em qualquer navegador.

Observe o código a seguir:

```
$('body p:nth-child(3)')
```

Aqui o construtor está usando um seletor CSS 3 que tem como alvo o terceiro parágrafo descendente do elemento `body`. Trata-se de uma busca simples, sem necessidade de atribuir um identificador ao terceiro parágrafo como se fez anteriormente.

A verdade é que jQuery usa amplamente seletores CSS e, assim sendo, é pré-requisito para bem desenvolver jQuery o conhecimento profundo desses seletores. No apêndice A, você encontra um guia de consulta aos seletores que irá ajudá-lo a acompanhar os exemplos deste livro.

1.1.7.2 Encadeamento jQuery

Um conceito importante da biblioteca jQuery é o encadeamento. Observe a linha de código a seguir:

```
$('div').children('p').fadeOut().addClass('xpto')
```

Não se preocupe se o código parecer confuso ou ininteligível, pois logo você estará entendendo-o. O código diz o seguinte:

> "Construtor, encontre todos os elementos parágrafos que sejam filhos dos elementos `div`, neles aplique um efeito de esmaecimento (fadeOut) e adicione a classe `xpto`".

Denomina-se encadeamento a característica de se poder encadear diversos métodos em uma declaração. Isso é possível porque se criou jQuery de modo a que cada método retorne um objeto pronto para receber outro método, que, por sua vez, retornará outro objeto, e assim por diante, permitindo ao desenvolvedor construir uma cadeia infinita de objetos e métodos. Em JavaScript tradicional, não existe encadeamento como o projetado para jQuery, o que obriga o uso de múltiplas declarações para se conseguir um efeito que, em jQuery, se consegue com uma linha de código apenas.

1.1.7.3 Funções utilitárias

Servir como inspetor e selecionador de componentes do DOM, conforme visto anteriormente, não é a única atribuição da função `$()`. jQuery prevê um conjunto de funções chamadas utilitárias que usa o sinal `$` como um identificador tal qual qualquer outro identificador previsto em JavaScript. A sintaxe para as funções utilitárias é mostrada no exemplo a seguir:

```
$.browser;
```

ou com a opção de uso do identificador jQuery:

```
jQuery.browser
```

O exemplo mostra uma função para identificar o tipo de navegador do usuário que equivale à função `navigator.userAgent` do JavaScript.

É muito pouco provável que você use uma função utilitária no desenvolvimento de scripts para funcionar em uma página web. Elas têm sua utilidade no desenvolvimento de extensões para a biblioteca jQuery, como a criação de plug-ins.

1.1.7.4 Conflitos com outras bibliotecas

Sem dúvida, a invenção de bibliotecas revigorou o uso de JavaScript no desenvolvimento de páginas web, pois além de tornar o processo de criação bem mais simples, forneceu mecanismos que possibilitam criar códigos não obstrutivos e, em consequência, sem prejuízos para usabilidade e acessibilidade da página. jQuery não detém exclusividade no campo das bibliotecas JavaScript, pelo contrário, muitas bibliotecas surgiram nos últimos tempos, entre elas: Prototype, MochiKit, MooTools, Yahoo User Interface (YUI) e DOMAssistant.

O identificador `$` também não é exclusividade de jQuery e outras bibliotecas fazem uso dele. Se um documento usa mais de uma biblioteca, é provável que ocorram conflitos com diferentes bibliotecas tentando interpretar um mesmo identificador e estabelecendo-se uma enorme confusão.

O sinal `$` é um pseudônimo e, no jargão técnico, diz-se "alias" para o identificador da biblioteca. O identificador utilizado foi jQuery, assim, ao usar essa biblioteca, `$` é o pseudônimo de `jQuery` e as duas sintaxes mostradas a seguir são equivalentes:

```
$()
```

e

```
jQuery()
```

Usar `jQuery()` elimina o risco de conflitos, mas obriga o desenvolvedor a abrir mão de escrever com a forma abreviada do pseudônimo.

Felizmente, para evitar conflitos com outras bibliotecas, sem prescindir de uma forma abreviada, existe a função `jQuery.noConflict()` que permite, entre outras funcionalidades, criar um pseudônimo personalizado para desenvolvimento. Tal função será abordada no capítulo 2, em [C2 S2.1].

CAPÍTULO 2

Funções-padrão e seletores jQuery

Na primeira parte deste capítulo, serão explicadas as funções-padrão, de emprego mais comum, da biblioteca jQuery, examinando a sintaxe, aplicação e funcionamento de objetos, métodos e propriedades que constituem o coração da biblioteca. Na sequência, serão examinados os poderosos seletores jQuery, apresentando-se sua sintaxe e desenvolvendo-se exemplos de aplicação.

2.1 Funções-padrão jQuery

Os arquivos exemplo contendo a demonstração prática das funções e seletores apresentados neste capítulo se baseiam na técnica de estilizar o(s) elemento(s)-alvo(s) de forma diferenciada no momento em que o script "roda" por ter sido acionado um disparador de eventos, em geral o clique em um botão.

Para adicionar estilos em uma seleção jQuery, pode-se usar duas sintaxes que serão estudadas com detalhes em [C4 S4.1], contudo veja a apresentação da sintaxe mais simples que será usada neste capítulo com o fim proposto no parágrafo anterior.

A sintaxe geral para adicionar estilo ao elemento-alvo de um seletorjQuery é mostrada a seguir:

```
seletorjQuery.css('propriedade', 'valor')
```

O parâmetro `propriedade` é uma propriedade CSS e o parâmetro `valor`, a quantificação ou característica da propriedade. Neste capítulo, serão usadas frequentemente as propriedades `backgroud` e `color` com os valores `red` (vermelho) e `green` (verde).

$(*expressão*, [*contexto*])

Esta é a principal função jQuery mencionada no capítulo 1, a qual aceita dois argumentos. Veja-os a seguir.

Argumento	Descrição
expressão	Um seletor CSS ou um dos seletores específicos da biblioteca.
contexto	O contexto em que será feita a procura do seletor. Omitindo esse argumento, o valor-padrão de procura são os elementos do DOM no documento atual. Você pode limitar a procura ao contexto de um elemento ou conjunto de elementos do DOM ou mesmo a um objeto jQuery.

Exemplo:

```
...
<head>
<style type="text/css" media="all">
   div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#diferente').css('background', 'red');     // sem uso do parâmetro contexto
      $('p', $('#container')).css('color', 'green'); // com uso do parâmetro contexto
   });
</script>
</head>
<body>
   <div></div>
   <div id="diferente"></div>
   <div></div>
<div id="container">
   <p>Parágrafo dentro do div#container</p>
</div>
   <p>Parágrafo fora do div#container</p>
...
```

[arquivo-2.1a.html]

Nesse script, verifica-se o uso do construtor `$()` sem o parâmetro *contexto*, quando se dá a busca pelo elemento cujo id é `diferente`, busca esta em todo o documento, para estilizar seu fundo na cor vermelha. Na outra situação, constata-se o uso do parâmetro *contexto* igual a `$('#container')`, quando se dá a busca pelos paragráfos dentro do `div#container`, para estilizá-los na cor verde. Se omitisse o parâmetro contexto nesse segundo caso, todos os parágrafos no documento seriam estilizados na cor verde.

$(*html*)

Esta função destina-se a criar marcação HTML. O valor do argumento passado é uma string contendo marcação estrutural que pode ser escrita manualmente em texto puro, usando um gerador de template, um plug-in ou AJAX. Neste livro, você verá a geração via texto puro escrito manualmente. Lembre-se de que criar marcação via JavaScript torna o conteúdo completamente inacessível a tecnologias assistivas. Use essa função consciente dessa limitação, lembrando que scripts jQuery destinam-se a um incremento progressivo do documento, portanto não crie marcação em desacordo com esse objetivo. Há limitações na criação de controles do tipo `input` para formulários com essa função, assim evite criá-los com o uso de script. Barras, quando presentes na string do argumento, devem ser escapadas.

Exemplo:

```
...
<head>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('<p>Eu sou um parágrafo criado com jQuery.</p> ').prependTo('body');
   });
</script>
</head>
<body>
</body>
</html>
```

[arquivo-2.1b.html]

Este script cria a marcação para um parágrafo e seu respectivo texto e usando o método `prependTo()`, que será apresentado adiante, escreve a marcação no elemento `body`. Se você visualizar o código-fonte da página, a marcação do parágrafo não estará lá e é isso que um leitor de tela encontra, ou seja, absolutamente nada.

$(*elementos*)

Esta função destina-se a procurar elementos HTML no DOM. Aceita também documentos XML e objetos `window`, ainda que não sejam componentes do DOM, como argumentos.

Exemplo:

```
...
<head>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div, p').css('background', 'green');
    });
</script>
</head>
<body>
    <div>DIV</div>
    <p>Parágrafo</p>
    <div>DIV</div>
</body>
</html>
```

[arquivo-2.1c.html]

$(callback)

Esta função destina-se a servir de container para scripts jQuery que devam ser executados somente após o carregamento do DOM. É uma função idêntica ao método `ready()` estudado em [C1 S1.1.6.2]. Tecnicamente, pode-se dizer que se trata de uma forma abreviada de escrever `$(document).ready()`.

Observe, no exemplo a seguir, que se trata do mesmo exemplo mostrado no item anterior, no qual se substituiu o método `ready()` por esta função.

Exemplo:

```
...
<head>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(function($) {
        $('div, p').css('background', 'green');
    });
</script>
</head>
<body>
<div>DIV</div>
<p>Parágrafo</p>
<div>DIV</div>
</body>
</html>
```

[arquivo-2.1d.html]

seletorjQuery.**each(callback)**

Usamos o termo *seletorjQuery* para designar genericamente um objeto ou array de objetos retornado por um simples seletor jQuery ou por um encadeamento jQuery.

Este método destina-se a executar uma função toda vez que encontra um elemento constante do conjunto procurado. Por exemplo: você deseja percorrer o DOM e mudar a cor de fundo de todos os divs do documento. Observe, a seguir, que se criou um botão para disparar o evento e percorreu-se o DOM usando o método `each()` para executar a mudança de cor de fundo.

Exemplo:

```
...
<style type="text/css" media="all">
   p {cursor:pointer; color:#00f;}
   div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').each(function(i) {
            $(this).css('background', 'red');
         });
      });
</script>
</head>
<body>
   <div>DIV</div>
   <div>DIV</div>
   <div>DIV</div>
   <button type="button">Colorir DIVS</button>
   ...
```

[arquivo-2.1e.html]

O script começa procurando o botão e a este atribui a tarefa de disparar a execução de uma função ao ser clicado. Tal função procura cada um dos divs no documento e estiliza-os com fundo na cor vermelha. Você deve estar se perguntando: por que complicar se basta declarar `$('div').css('background', 'red')`? Certo, essa simples declaração teria o mesmo efeito, contudo note que existe um parâmetro *i* na função definida em `each()`. Esse parâmetro retorna um contador dos elementos encontrados. Essa função funciona como um *loop for* do JavaScript. No arquivo de exemplo, acrescentou-se um `alert(i)` para você acompanhar o funcionamento da função.

seletorjQuery.length

Esta propriedade retorna o número de ocorrências do elemento-alvo.

Exemplo:

```
...
<style type="text/css" media="all">
    p {cursor:pointer;}
    div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('p').click(function() {
            var nDiv = $('div').length;
            $('<span>Foram encontradas ' + nDiv + ' DIVs.</span>').appendTo('p');
        });
    });
</script>
</head>
<body>
    <p>Clique aqui para saber quantos divs existem nesse documento.<br /></p>
        <div>DIV</div>
        <div>DIV</div>
        <div>DIV</div>
    ...
```

[arquivo-2.1f.html]

seletorjQuery.eq(*posição*)

Esta função retorna uma determinada ocorrência do conjunto de elementos-alvo. O argumento `posição` refere-se à posição da ocorrência no conjunto selecionado. No exemplo a seguir, selecionou-se o terceiro item da lista. Não esqueça que em JavaScript a contagem começa em 0 (zero).

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
    $('li').eq(2).css('color', 'red')
    )};
</head>
```

```
<body>
   <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
      <li>Item 4</li>
      <li>Item 5</li>
   </ul>
   ...
```

[arquivo-2.1g.html]

seletorjQuery.get()

Esta função destina-se a acessar elementos do DOM sem emprego das funcionalidades jQuery. Pode ser usada para manipular diretamente o DOM e foi criada para resolver problemas de retro-compatibilidade. Retorna um array de elementos.

Exemplo:

```
...
<style type="text/css" media="all">
   div {margin:20px 0; color:red;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      function conteudoParagrafos(p) {
         var arrayConteudos = [ ];
         for (var i = 0; i < p.length; i++) {
            arrayConteudos.push(p[i].innerHTML);
         }
         $('div').text(arrayConteudos.join(' - '));
      }

      $('button').click(function() {
         conteudoParagrafos($('p').get().reverse());
      });

      $('.reset').click(function() {
         $('div').empty();
      });
   });
</script>
</head>
```

```
<body>
    <button type="button">Rodar script</button>
    <button type="button" class="reset">Reset</button>
    <p>Parágrafo um.</p>
    <p>Parágrafo dois.</p>
    <p>Parágrafo três.</p>
    <div></div>
...
```

[arquivo-2.1h.html]

seletorjQuery.get(*índice*)

Esta função tem a mesma finalidade da anterior e destina-se a acessar um elemento específico do conjunto de elementos retornado. O parâmetro `índice` indica a posição do elemento na array retornada.

seletorjQuery.index(*alvo*)

Esta função retorna o índice do elemento definido no parâmetro `alvo`. A contagem começa em 0 (zero) e, caso não exista um índice para o alvo, a função retorna o valor -1. No exemplo a seguir, o script percorre e encontra todos os parágrafos do documento e recolhe seus índices. A demonstração, no exemplo, se faz com um clique em cada um dos parágrafos.

Exemplo:

```
...
<style type="text/css" media="all">
    div {margin:20px 0;}
    span {color:red;}
    p {cursor:pointer;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('p').click(function() {
            var indice = $('p').index(this);
            $('div').append('O índice deste parágrafo é: <span>' + indice + '</span><br />');
        });
        $('.reset').click(function() {
            $('div').empty();
        });
    });
</script>
</head>
```

```
<body>
   <button type="button" class="reset">Reset</button><br />
   <b>Clique em cada um dos parágrafos abaixo <br />
   para revelar seu índice</b>
   <p>Parágrafo um.</p>
   <p>Parágrafo dois.</p>
   <p>Parágrafo três.</p>
   <div></div>
...
```

[arquivo-2.1i.html]

jQuery.noConflict()

Esta função é utilizada para evitar conflitos da biblioteca jQuery com outras bibliotecas usadas no mesmo documento e que fazem uso do *alias* `$`.

Existem várias bibliotecas JavaScript disponíveis e, se você optou por usar JQuery, sem dúvida fez uma escolha inteligente e não estará impedido de utilizar outras bibliotecas em um mesmo projeto, desde que tome alguns cuidados para evitar conflitos.

Conflitos ocorrem porque diferentes bibliotecas, com seus diversos métodos, usando uma sintaxe comum para invocar suas funcionalidades (o alias `$` ou o construtor `$()`), acabam por permitir que as diferentes bibliotecas tentem interpretar o mesmo código, criando um caos como este: a biblioteca A tentando interpretar uma função criada para a biblioteca B e vice-versa.

Existem diferentes técnicas para evitar conflitos e a mais simples é escrever códigos usando o prefixo jQuery no lugar do alias `$`. Essa solução, apesar de simples, tem a desvantagem de aumentar o número de caracteres a digitar e é fácil perceber que irá exigir trabalho extra, principalmente em scripts extensos. A sintaxe geral para essa solução é mostrada a seguir:

```
...
<script src="outra_biblioteca.js"></script>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   jQuery.noConflict();
   // Script para biblioteca jQuery usando jQuery()
   jQuery(document).ready(function() {
      jQuery('p').show();
      jQuery('div').css('color', 'red');
      jQuery(...;
      ...
   });
```

```
    // Segue script para outra biblioteca usando alias $()
    $('#nome-id').show();
    $(...;
</script>
...
```

A segunda solução consiste em personalizar um *alias* para uso próprio. Graças a funcionalidades nativas da biblioteca, você pode criar o nome que bem entender para substituir o *alias* nativo `$`. Assim, é possível adotar sintaxe tais como as mostradas a seguir:

```
$maujor('div').hide();
$carlosalberto('p').show();
$j('span').fadeIn();
```

O primeiro e segundo exemplos não são uma boa escolha para um *alias*, pois deve-se procurar simplicidade e redução de digitação. O último exemplo é bem melhor e o mais simples possível, pois se escolheu uma letra apenas para *alias*. Veja a seguir a sintaxe para sua criação:

```
...
<script src="outra_biblioteca.js"></script>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    var $j = jQuery.noConflict();
    // Script para biblioteca jQuery usando $j()
    $j(document).ready(function() {
        $j('p').show();
        $j('div').css('color', 'red');
        $j(...;
        ...
    });
    // Segue script para outra biblioteca usando alias $()
    $('#nome-id').show();
    $(...;
</script>
...
```

A terceira solução permite-lhe usar o *alias* `$` tanto para a biblioteca jQuery como para outra biblioteca. Nessa solução, você cria um script que funciona como uma espécie de container para seus códigos jQuery e insere os scripts da outra biblioteca fora desse container. O container nada mais é que uma função definindo `$` como jQuery. O exemplo a seguir esclarece essa solução:

```
...
<script src="outra_biblioteca.js"></script>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   jQuery.noConflict();
   $(document).ready(function() {
      (function($) {
         // Script para biblioteca jQuery usando $()
         $('p').show();
         $('div').css('color', 'red');
         $(...;
         ...
      }) (jQuery);
   )};
   // Segue script para outra biblioteca usando alias $()
   $('#nome-id').show();
   $(...;
</script>
...
```

2.2 Seletores jQuery

Para um sólido entendimento de seletores, é necessário que você esteja familiarizado com a árvore do documento e o consequente relacionamento entre os elementos que a compõem. Não é sem razão que se emprega o termo árvore do documento e a melhor maneira de se entender seu funcionamento e relacionamentos é fazer analogia com a conhecida árvore genealógica examinando o grau de parentesco entre os elementos de um documento.

A terminologia técnica de referência aos inter-relacionamentos na árvore do documento utiliza os mesmos termos usados em árvore genealógica. Na árvore de um documento, existem elementos-pai, elementos ancestrais, elementos-filho, elementos-irmão e assim por diante.

Considere um documento com a seguinte marcação:

```
...
<body>
   <div id="topo">
      <h1>empresa</h1>
      <ul id="nav">
         <li><a href="#">Link 1</a></li>
         <li><a href="#">Link 2</a></li>
         <li><a href="#">Link 3</a></li>
      </ul>
   </div>
```

```
        <div id="principal">
            <h2>Título</h2>
            <p>...parágrafo <em>itálico</em>...</p>
            <h2>Título</h2>
            <blockquote><p>...citação...</p></blockquote>
        </div>
    <hr />
    </body>
    </html>
```

Veja, na figura 2.1, o diagrama representativo da árvore do documento.

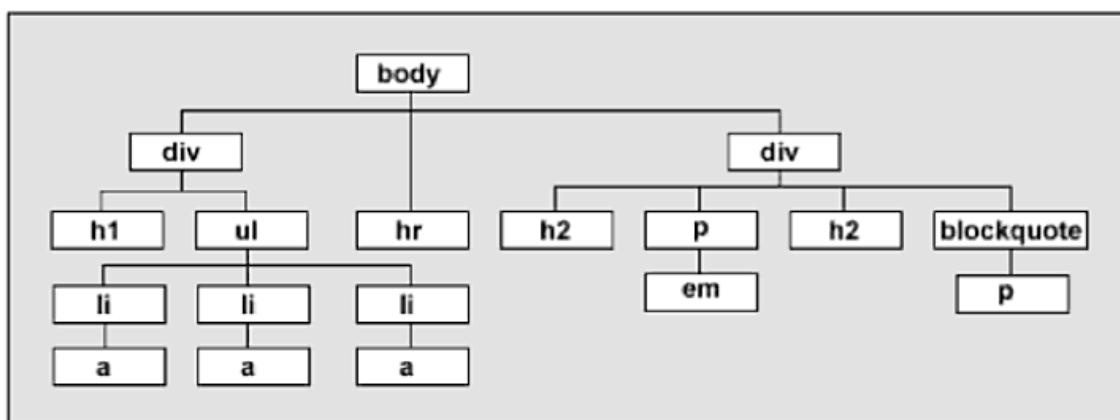



Figura 2.1 – Árvore do documento.

Veja alguns exemplos da terminologia que se aplica aos elementos mostrados na árvore do documento:

- `body` é *ancestral* de todos os elementos. Todos os elementos são *descendentes* de `body`.
- `ul` é *ancestral* de `li` e de `a`. `li` e `a` são *descendentes* de `ul`.
- `li` é *elemento-filho* de `ul`. `ul` é *elemento-pai* de `li`.
- `blockquote` é *irmão* de `p` e também tem um filho `p`.
- `blockquote` é *irmão adjacente* de `h2`
- `em` e `p` não são *irmãos*.
- `li` não é *filho* de `div`, mas é seu *descendente*.

A tabela 2.1 mostra como o relacionamento entre elementos na árvore do documento determina a sintaxe do seletor. Os termos serão frequentemente citados neste capítulo e seu entendimento é fundamental para a compreensão do funcionamento dos seletores.

Tabela 2.1 – Elementos na árvore do documento

Termo	Relacionamento	Sintaxe adotada
Elemento-pai	O ascendente direto do elemento	*pai* > *filho*
Elemento-filho	O descendente direto do elemento	*pai* > *filho*
Elemento-irmão	Elementos-filho de um mesmo pai	
Elemento adjacente	Elemento que se segue a outro	A + B
Elemento-irmão adjacente	Elementos-filho do mesmo pai	A ~ B

2.2.1 Seletores simples

$(*id*)

Seletor de `id`: acessa o elemento cujo valor do atributo *`id`* tenha sido especificado no argumento.

Exemplo:

```
...
<style type="text/css" media="all">
    div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('#diferente').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
        <div>DIV</div>
        <div>DIV</div>
        <div id="diferente">DIV</div>
...
```

[arquivo-2.2.1a.html]

Nesse exemplo, utilizou-se o seletor `id` `$('#diferente')` para acessar o `div` ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

$(*classe*)

Seletor de classe: acessa os elementos cujo valor do atributo `class` tenha sido especificado no argumento.

Exemplo:

```
...
<style type="text/css" media="all">
   div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('.diferente').css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
      <div class="diferente">DIV</div>
      <div class="diferente">DIV</div>
      <div>DIV</div>
...
```

[arquivo-2.2.1b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor de classe `$('.diferente')` para acessar os dois divs, os quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

$(*elemento*)

Seletor de elementos: acessa todos os elementos especificados no argumento.

Exemplo:

```
...
<style type="text/css" media="all">
   div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
```

```
            $('div').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <div>DIV</div>
    <div>DIV</div>
    <div>DIV</div>
...
```

[arquivo-2.2.1c.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor de elemento `$('div')` para acessar todos os divs existentes no documento, os quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

$(*seletor1*, *seletor2*, *seletor3*, ...)

Grupamento de seletores: acessa um agrupamento de seletores. O argumento é uma lista dos seletores a acessar.

Exemplo:

```
...
<style type="text/css" media="all">
    div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('p, .diferente, li').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
        <div>DIV</div>
        <p>Texto de um parágrafo</p>
        <div class="diferente">DIV com class="diferente"</div>
        <p class="outra"> Parágrafo com class="outra"</p>
```

```
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
...
```

[arquivo-2.2.1d.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o grupo de seletores `p`, `.diferente` e `li` para acessar todas as instâncias dos elementos às quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Note que o parágrafo ao qual se atribuiu a classe `outra` é alvo do seletor e foi estilizado, pois a presença do atributo `class` não altera a condição do elemento de ser um parágrafo.

2.2.2 Seletores compostos

Seletor composto é aquele constituído pela combinação de dois ou mais seletores simples. A combinação entre seletores simples para criar um seletor composto segue uma sintaxe própria e é feita com o emprego de três sinais de combinação como descrito na tabela 2.2.

Tabela 2.2 – Sinais de combinação

Sinal de combinação	Exemplo ilustrativo	Nota
Espaço em branco	div p em	Obrigatório usar um ou mais espaços entre seletores
Sinal de maior ">"	div > p ou div>p	Espaçamento facultativo entre seletores
Sinal de adição "+"	p + a ou p+a	Espaçamento facultativo entre seletores

$(*ancestral descendente*)

Acessa o elemento descendente do ancestral especificado no argumento.

Exemplo:

```
...
<style type="text/css" media="all">
    div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
```

```
    $(document).ready(function() {
        $('button').click(function() {
            $('div span').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <div>
    <p>Parágrafo filho do div com <span>texto</span> marcado com span</p>
    </div>
    <p>Parágrafo filho de body com <span>texto</span> marcado com span </p>
...
```

[arquivo-2.2.2a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor ancestral descendente para acessar os elementos `span` descendentes do `div` (neste exemplo, apenas um elemento `span`) aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Note que os dois elementos `span` constantes do DOM são filhos de um parágrafo, mas somente o primeiro é descendente do `div`.

$(*pai* > *filho*)

Acessa o elemento-filho do pai especificado no argumento.

Exemplo:

```
...
<style type="text/css" media="all">
    div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('div > span').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
```

```
    <div>
    <p>Parágrafo filho do div com <span>texto</span> marcado com span</p>
    </div>
    <p>Parágrafo filho de body com <span>texto</span> marcado com span </p>
...
```

 [arquivo-2.2.2b.html]

Utilizaram-se um botão para disparar o evento e o seletor-pai-filho para acessar os elementos `span` filho do `div` (nesse exemplo, há duas ocorrências do elemento `span`) aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Observe que a marcação usada nesse exemplo é a mesma do exemplo anterior, no entanto como não há elemento `span` filho do `div` – são filhos dos parágrafos –, nada acontece quando se clica o botão. Caso tivesse usado o seletor `$('p > span')`, os dois elementos `span` seriam alvo do seletor. Faça uma cópia do arquivo e altere o seletor como sugerido.

$(*anterior* + *próximo*)

Acessa o elemento *`próximo`* que se segue imediatamente ao elemento *`anterior`*.

Exemplo:

```
...
<style type="text/css" media="all">
    div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('div + p').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <p>Parágrafo antes do div</p>
    <div>
    <p>Parágrafo filho do div</p>
    </div>
    <p>Parágrafo que se segue ao div</p>
    <p>Parágrafo que seguinte</p>
...
```

 [arquivo-2.2.2c.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor anterior próximo para acessar o elemento `p` imediatamente após o `div` ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

$(*anterior ~ irmãos*)

Este é um seletor previsto nas CSS 3 que acessa todos os elementos *`irmãos`* e que se seguem ao elemento *`anterior`*.

Exemplo:

```
...
<style type="text/css" media="all">
   div {width:200px; height:100px; border:1px solid #000; margin:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('h2 ~ p').css('background', 'red');
      });
   });
</script>
</head>
<body>

<button type="button">Vermelha</button>
<p>Parágrafo antes do cabeçalho h2</p>
<h2>Cabeçalho h2</h2>
<p>Primeiro parágrafo após o cabeçalho h2</p>
<p>Segundo parágrafo após o cabeçalho h2</p>
<div>DIV</div>
<p>Terceiro parágrafo após o cabeçalho h2</p>

<div>
<p>Parágrafo dentro de um div</p>
</div>
...
```

[arquivo-2.2.2d.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor *`anterior irmão`* para acessar todos os elementos `p` irmão que estão após o cabeçalho `h2` e aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

Note o seguinte:

- O parágrafo antes do cabeçalho h2 não é selecionado, pois o seletor é para elementos posteriores a h2.
- O primeiro e segundo parágrafos após h2 são selecionados porque são filhos do mesmo pai (o elemento body), portanto irmãos.
- O elemento div que se segue não é selecionado, pois o seletor é para parágrafos.
- O parágrafo que se segue ao div é selecionado, pois é filho de body, está após h2 e, portanto, é irmão dos parágrafos anteriores.
- O parágrafo dentro do div que se segue não é selecionado, pois não sendo filho de body (é filho de um div), não é irmão dos parágrafos anteriores.

2.2.3 Seletores – Filtros básicos

Denominam-se seletores filtros básicos os seletores do tipo pseudosseletores que se destinam a filtrar uma condição particular de um seletor simples ou composto.

seletor:first

Acessa a primeira ocorrência do conjunto de elementos selecionados por *seletor*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#btn-vermelha').click(function() {
            $ ('li:first').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <ul>
        <li>Primeiro item</li>
```

```
        <li>Segundo item</li>
        <li>Terceiro item</li>
        <li>Quarto item</li>
        <li>Quinto item</li>
    </ul>
...
```

 [arquivo-2.2.3a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:first` para acessar o primeiro item da lista ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor:last

Acessa a última ocorrência do conjunto de elementos selecionados por *`seletor`*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#btn-vermelha').click(function() {
            $('li:last').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <ul>
        <li>Primeiro item</li>
        <li>Segundo item</li>
        <li>Terceiro item</li>
        <li>Quarto item</li>
        <li>Quinto item</li>
    </ul>
...
```

[arquivo-2.2.3b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e usamos o seletor `:last` para acessar o último item da lista ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

***seletor1*:not(*seletor2*)**

Este é um seletor previsto nas CSS 3 que acessa o conjunto de seletores *seletor1*, excluindo as instâncias definidas em *seletor2*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('li:not(li:first)').css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <ul>
      <li>Primeiro item</li>
      <li>Segundo item</li>
      <li>Terceiro item</li>
      <li>Quarto item</li>
      <li>Quinto item</li>
   </ul>
...
```

[arquivo-2.2.3c.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento, o seletor `li` para acessar todos os elementos itens de lista e o seletor `:not` para excluir o primeiro item da lista ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Todos os itens da lista serão selecionados, menos o primeiro (`not(li:first)`).

seletor:even

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa as ocorrências de ordem par do conjunto de elementos selecionados por `seletor`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('tr:even').css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <table border="1">
   <tr>
      <td>Célula índice 0</td>
   </tr>
   <tr>
      <td>Célula índice 1</td>
   </tr>
      <tr>
      <td>Célula índice 2</td>
   </tr>
   <tr>
      <td>Célula índice 3</td>
   </tr>
   <tr>
      <td>Célula índice 4</td>
   </tr>
   </table>
...
```

[arquivo-2.2.3d.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `tr:even` para acessar todas as linhas pares da tabela às quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Lembre-se de que, na linguagem JavaScript, toda contagem começa em 0 (zero) e, em consequência, as ocorrências pares são a primeira (índice 0, que é contado como seleção par), a terceira (índice 2), a quinta (índice 4), e assim por diante.

seletor:odd

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa as ocorrências de ordem ímpar do conjunto de elementos selecionados por `seletor`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('tr:odd').css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <table border="1">
   <tr>
      <td>Célula índice 0</td>
   </tr>
   <tr>
      <td>Célula índice 1</td>
   </tr>
      <tr>
      <td>Célula índice 2</td>
   </tr>
   <tr>
      <td>Célula índice 3</td>
   </tr>
   <tr>
      <td>Célula índice 4</td>
   </tr>
   </table>
...
```

[arquivo-2.2.3e.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `tr:odd` para acessar todas as linhas ímpares da tabela às quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Lembre-se de que, na linguagem JavaScript, toda contagem começa em 0 (zero) e, em consequência, as ocorrências ímpares são a segunda (índice 1), a quarta (índice 3), a sexta (índice 5), e assim por diante.

seletor:eq(*índice*)

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa a ocorrência de ordem `índice` no conjunto de elementos selecionados por `seletor`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('td:eq(4)').css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <table border="1">
      <tr>
         <td>td 0</td>
         <td>td 1</td>
      </tr>
      <tr>
         <td>td 2</td>
         <td>td 3</td>
      </tr>
      <tr>
         <td>td 4</td>
         <td>td 5</td>
      </tr>
      <tr>
         <td>td 6</td>
         <td>td 7</td>
      </tr>
   </table>
...
```

[arquivo-2.2.3f.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `td:eq(4)` para acessar a célula da tabela que ocupa a quarta posição no conjunto das células selecionadas à qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Lembre-se de que, na linguagem JavaScript, toda contagem começa em 0 (zero) e, em consequência, a quarta ocorrência corresponde à quinta célula.

***seletor*:gt(*índice*)**

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências do conjunto de elementos selecionados por `seletor` cujo índice é maior que o índice definido no parâmetro `índice`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#btn-vermelha').click(function() {
            $('td:gt(4)').css('background', 'red');
        });
    });
</script>
</head>
<body>

    <button type="button">Vermelha</button>
    <table border="1">
        <tr>
            <td>td 0</td>
            <td>td 1</td>
        </tr>
        <tr>
            <td>td 2</td>
            <td>td 3</td>
        </tr>
        <tr>
            <td>td 4</td>
            <td>td 5</td>
        </tr>
        <tr>
            <td>td 6</td>
            <td>td 7</td>
    </tr>
    </table>
...
```

[arquivo-2.2.3g.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `td:gt(4)` para acessar as células cujo índice é maior que quatro no conjunto das células selecionadas, às quais será anexado o comportamento definido no script (mudar

a cor de fundo para vermelha). Lembre-se de que, na linguagem JavaScript, toda contagem começa em 0 (zero) e, em consequência, a quarta ocorrência corresponde à quinta célula.

seletor:**lt**(*índice*)

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências do conjunto de elementos selecionados por `seletor` cujo índice é menor que o índice definido no parâmetro `índice`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#btn-vermelha').click(function() {
            $('td:lt(4)').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <table border="1">
        <tr>
            <td>td 0</td>
            <td>td 1</td>
        </tr>
        <tr>
            <td>td 2</td>
            <td>td 3</td>
        </tr>
        <tr>
            <td>td 4</td>
            <td>td 5</td>
        </tr>
        <tr>
            <td>td 6</td>
            <td>td 7</td>
        </tr>
    </table>
...
```

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `td:lt(4)` para acessar as células cujo índice é menor que quatro no conjunto das células selecionadas, às quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Lembre de que, na linguagem JavaScript, toda contagem começa em 0 (zero) e, em consequência, a quarta ocorrência corresponde à quinta célula.

$(*:header*)

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências de cabeçalhos de qualquer nível, ou seja, todos os elementos `h1` até `h6`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#btn-vermelha').click(function() {
            $(':header').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <h1>cabeçalho nível 1</h1>
        <p>texto de um parágrafo</p>
    <h2>cabeçalho nível 2</h2>
        <p>texto de um parágrafo</p>
    <h3>cabeçalho nível 3</h3>
        <p>texto de um parágrafo</p>
    <h4>cabeçalho nível 4</h4>
...
```

[arquivo-2.2.3i.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e usamos o seletor `:header` para acessar todos os cabeçalhos, aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor:animated

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa as ocorrências do elemento `seletor` para as quais se tenha definido um script de animação. Esse seletor será visto com mais detalhes ao se abordarem as técnicas jQuery de animação.

2.2.4 Seletores de conteúdo

seletor:contains(*texto*)

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências de `seletor` que contenham um texto (string) definido no parâmetro `texto`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('p:contains('Maujor').css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <p>Visite o site do Maujor</p>
   <p>Estude CSS e jQuery</p>
   <p>HTML 5 está em fase de estudos</p>
   <p>Maujor escreveu este livro</p>
   <span>Elemento span contendo Maujor</span>
   <p>parágrafo contendo maujor</p>
...
```

[arquivo-2.2.4a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor :`contais` para acessar todos os elementos `p` que contêm a palavra Maujor, aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Note que não se selecionaram o elemento `span` (não é parágrafo) nem o elemento `p` contendo maujor (minúscula), pois o seletor é sensível ao tamanho da caixa da string passada pelo parâmetro `texto`.

seletor:empty

Este é um seletor previsto nas CSS 3 que acessa todas as ocorrências vazias de *seletor*.

Exemplo:

```
...
<style type="text/css" media="all">
   table tr td {height:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('td:empty.css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <table border="1">
   <tr>
      <td>Célula índice 0</td>
   </tr>
   <tr>
      <td></td>
   </tr>
      <tr>
      <td>Célula índice 2</td>
   </tr>
   <tr>
      <td>Célula índice 3</td>
   </tr>
   <tr>
      <td></td>
   </tr>
   </table>
...
```

[arquivo-2.2.4b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `td:empty` para acessar todas as células da tabela que estão vazias, às quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor1:**has**(*seletor2*)

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências do elemento definido pelo `seletor1` que contenham pelo menos uma ocorrência do elemento definido pelo `seletor2`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('p:has(strong).css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <p>Texto com <i>palavra</i> em itálico</p>
   <p>Texto com <b>palavra</b> em negrito</p>
   <p>Texto com <span>palavra</span> dentro de span</p>
   <p>Texto com <strong>palavra</strong> com forte ênfase</p>
   <p>Texto sem marcação extra</p>
...
```

[arquivo-2.2.4c.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `p:has(strong)` para acessar todas as ocorrências de parágrafos que contenham o elemento `strong`, aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

Note que enquanto o seletor `:contains()` estudado anteriormente selecionou elementos que contêm um determinado texto, este seletor `:has()` selecionou elementos que contêm um outro determinado elemento.

seletor:**parent**

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências de elementos que sejam elementos-pai, ou seja, elementos que tenham filhos, inclusive texto (text-node) como filho.

Exemplo:

```
...
<style type="text/css" media="all">
   p {height:20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('#btn-vermelha').click(function() {
         $('p:parent).css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <p>Texto com <i>palavra</i> em itálico</p>
   <p></p>
   <p>Texto sem marcação extra</p>
...
```

[arquivo-2.2.4d.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `p:parent` para acessar todas as ocorrências de parágrafos que contêm elementos-filho às quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). Observe que na marcação há um elemento parágrafo vazio e que nas CSS se definiu uma altura de 20px para os parágrafos. Esse elemento não é alvo de seu seletor. Para entender o funcionamento desse seletor, faça uma cópia do arquivo exemplo e altere o seletor para `$('p')`, pois assim notará que na visualização de sua cópia, no funcionamento do script, aparece o parágrafo vazio com fundo na cor vermelha.

2.2.5 Seletores de visibilidade

:hidden

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências de elementos que estejam ocultos no documento, incluindo campos de formulário `input` do tipo `hidden`.

Normalmente, este seletor acessa também os elementos de marcação contidos na seção head do documento, portanto se esta não é sua intenção, limite a busca à seção body com a seguinte sintaxe: `$(':hidden', $('body'))`

Exemplo:

```
...
<style type="text/css" media="all">
   h1 {display:hidden;}    /* esconde o elemento h1 */
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var totalOcultos = $(':hidden', $('body')).length;
         var inputOcultos = $('input:hidden').length;
         $('span').text('Foram encontrados ' + totalOcultos +
            ' elementos ocultos, sendo ' + inputOcultos +
            ' elementos input do tipo hidden.')      /* escrever o resultado */
      });
   });
</script>
</head>
<body>
   <button type="button">Ocultos</button>
   <h1>Cabeçalho oculto por declaração CSS</h1>
   <form>
      <input type="hidden" />
      <input type="hidden" />
   </form>
   <span></span> <!--aqui jQuery escreve os resultados de elementos ocultos encontrados -->
   ...
```

[arquivo-2.2.5a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor :hidden duas vezes: a primeira para acessar o total de ocorrências de elementos ocultos e a segunda para acessar a ocorrência de elementos `input` do tipo `hidden`. Os números de ocorrências foram passados a variáveis e escritos no documento para constatação do funcionamento do script.

Note que elementos ocultos pelas declarações CSS `display:none` (como mostrado no exemplo) ou `visibility:hidden` são acessados pelo seletor.

Esse seletor admite as seguintes sintaxes:

Sintaxe	Descrição
`$(':hidden')`	Acessa todos os elementos ocultos no documento, incluindo os elementos constantes da seção `head`, tais como `title`, `style`, `scripts`, `meta` etc.

Sintaxe	Descrição (cont.)
`$(':hidden', $('body'))`	Acessa todos os elementos ocultos no documento que constem da seção `body`. Veja `jQuery(expressão, [contexto])` em [C2 S2.1].
`$('seletor:hidden')`	Acessa todos os elementos do tipo *seletor*, ocultos no documento.

Sugestão: faça uma cópia do arquivo de exemplo e retire a seleção contextual `$('body')` do seletor. Você irá notar que o número de elementos ocultos retornados aumentou, pois na visualização de sua cópia alterada, no funcionamento do script, foram incluídos os elementos da seção `head`.

:visible

Este é um seletor não previsto nas CSS e exclusivo da biblioteca, que acessa todas as ocorrências de elementos visíveis (não ocultos) no documento.

Normalmente, este seletor acessa também os elementos de marcação contidos na seção head do documento, portanto se esta não é sua intenção, limite a busca à seção body com a seguinte sintaxe: `$(':visible', $('body'))`

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
      var totalVisiveis = $(':visible', $('body')).length;
      $('span').text('Foram encontrados ' + totalVisiveis +
         ' elementos visíveis no documento.')     /* escrever o resultado */
      });
   });
</script>
</head>
<body>
   <button type="button">Ocultos</button>
   <h1>Cabeçalho de nível 1</h1>
   <p>Texto de um parágrafo</p>
   <span></span> <!--aqui jQuery escreve o resultado de elementos visiveis encontrados -->
   ...
```

[arquivo-2.2.5b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e usamos o seletor `:visible` para acessar o total de ocorrências de elementos visíveis na seção `body` do documento. Os resultados foram passados a uma variável e escritos no documento para constatação do funcionamento do script.

Note que elementos ocultos pelas declarações CSS `display:none` ou `visibility: hidden` não são acessados pelo seletor.

Esse seletor admite as seguintes sintaxes:

Sintaxe	Descrição
`$(':visible')`	Acessa todos os elementos visíveis no documento, incluindo os elementos constantes da seção `head`, tais como `title`, `style`, `scripts`, `meta` etc.
`$(':visible', $('body'))`	Acessa todos os elementos visíveis no documento que constem da seção `body`.
`$('`*`seletor`*`:visible')`	Acessa todos os elementos do tipo *`seletor`*, visíveis no documento.

2.2.6 Seletores de atributo

Os seletores de atributo são previstos nas CSS 3.

seletor[*atributo*]

Acessa todas as ocorrências do elemento *`seletor`* para o qual se tenha declarado o atributo definido no parâmetro *`atributo`*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p[title]'). css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <h3>Cabeçalho nível 3</h3>
   <p>Texto de um parágrafo <b>sem</b> atributo title.<p>
   <p title="meu_titulo">Texto de um parágrafo <b>com</b> atributo title.</p>
   <p lang="pt-br">Texto de um parágrafo com atributo lang.</p>
   <p title="outro_titulo">Texto de um parágrafo com atributo title.</p>
...
```

[arquivo-2.2.6a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor p[title] para acessar os elementos parágrafo que contêm o atributo title aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor[atributo = "valor"]

Acessa todas as ocorrências do elemento *seletor* para o qual se tenha declarado o par *atributo = "valor"*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p[title = "meu_titulo"]'). css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <h3>Cabeçalho nível 3</h3>
   <p>Texto de um parágrafo <b>sem</b> atributo title = "meu_titulo".</p>
   <p title="meu_titulo">Texto de um parágrafo <b>com</b> atributo
      title = "meu_titulo".</p>
   <p lang="pt-br">Texto de um parágrafo com atributo lang.</p>
   <p title="outro_titulo">Texto de um parágrafo com atributo
      title = "outro_titulo".</p>
...
```

[arquivo-2.2.6b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor p[title = "meu_titulo"] para acessar os elementos parágrafo que contêm o atributo title com o valor "meu_titulo" aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor[*atributo* != "*valor*"]

Acessa todas as ocorrências do elemento `seletor` para o qual o valor do atributo declarado em `atributo` não seja `valor`. Esse seletor é a negação do seletor anterior, ou seja, acessa todos os elementos definidos em `seletor` que não sejam acessados pelo seletor estudado anteriormente.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p[title != "meu_titulo"]'). css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <h3>Cabeçalho nível 3</h3>
   <p>Texto de um parágrafo <b>sem</b> atributo title = "meu_titulo".</p>
   <p title="meu_titulo">Texto de um parágrafo <b>com</b> atributo
      title = "meu_titulo".</p>
   <p lang="pt-br">Texto de um parágrafo com atributo lang.</p>
   <p title="outro_titulo">Texto de um parágrafo com atributo
      title = "outro_titulo".</p>
...
```

[arquivo-2.2.6c.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `p[title != "meu_titulo"]` para acessar os elementos parágrafo que não contêm o atributo `title` com o valor `"meu_titulo"` aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

Note que o seletor de negação não procurou somente os elementos que possuem o atributo negado. Caso o elemento não possua o atributo, é considerado alvo do seletor. Ao usar o seletor de negação, é mais seguro raciocinar com a seleção complementar da seleção negada.

seletor[atributo ^="valor"]

Acessa todas as ocorrências do elemento `seletor` para o qual o valor do atributo declarado em `atributo` começa com a string `valor`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p[title ^= "out"]'). css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <h3>Cabeçalho nível 3</h3>
   <p>Texto de um parágrafo <b>sem</b> atributo title = "meu_titulo".</p>
   <p title="meu_titulo">Texto de um parágrafo <b>com</b> atributo
      title = "meu_titulo".</p>
   <p lang="pt-br">Texto de um parágrafo com atributo lang.</p>
   <p title="outro_titulo">Texto de um parágrafo com atributo
      title = "outro_titulo".</p>
...
```

[arquivo-2.2.6d.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `p[title ^= "out"]` para acessar os elementos parágrafo que contêm o valor do atributo `title` começando com a string `"out"` aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha). O acesso será ao parágrafo com o atributo `title` de valor igual a `outro_titulo`.

seletor[atributo $="valor"]

Acessa todas as ocorrências do elemento `seletor` para o qual o valor do atributo declarado em `atributo` termina com a string `valor`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
```

```
    $(document).ready(function() {
        $('button').click(function() {
            $('p[title $= "tulo"]'). css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <h3>Cabeçalho nível 3</h3>
    <p>Texto de um parágrafo <b>sem</b> atributo title = "meu_titulo".</p>
    <p title="meu_titulo">Texto de um parágrafo <b>com</b> atributo
        title = "meu_titulo".</p>
    <p lang="pt-br">Texto de um parágrafo com atributo lang.</p>
    <p title="outro_titulo">Texto de um parágrafo com atributo
        title = "outro_titulo".</p>
...
```

[arquivo-2.2.6e.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `p[title $= "tulo"]` para acessar os elementos parágrafo que contêm o valor do atributo `title` terminado com a string `"tulo"` aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor[*atributo* *= "*valor*"]

Acessa todas as ocorrências do elemento *`seletor`* para o qual o valor do atributo declarado em *`atributo`* contém a string *`valor`*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('p[title *= "_tit"]'). css('background', 'red');
        });
    });
</script>
</head>
<body>

    <button type="button">Vermelha</button>
    <h3>Cabeçalho nível 3</h3>
```

```
    <p>Texto de um parágrafo <b>sem</b> atributo title = "meu_titulo".</p>
    <p title="meu_titulo">Texto de um parágrafo <b>com</b> atributo
       title = "meu_titulo".</p>
    <p lang="pt-br">Texto de um parágrafo com atributo lang.</p>
    <p title="outro_titulo">Texto de um parágrafo com atributo
       title = "outro_titulo".</p>
...
```

 [arquivo-2.2.6f.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `p[title *= "_tit"]` para acessar os elementos parágrafo que contêm no valor do atributo `title` a string `_tit` aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

***seletor*[*filtro1 atributo*]...[*filtroN atributo*]**

Acessa todas as ocorrências do elemento *`seletor`* para as quais os atributos satisfazem todas as condições declaradas nos filtros de atributos. Você pode usar quantos filtros quiser.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p[title $= "lo"][id]').css('background', 'red');
      });
   });
</script>
</head>
<body>
   <button type="button">Vermelha</button>
   <h3>Cabeçalho nível 3</h3>
   <p>Texto de um parágrafo <b>sem</b> atributo title = "meu_titulo".</p>
   <p title="meu_titulo" id="um">
      Texto de um parágrafo <b>com</b> atributo title = "meu_titulo" e atributo id.
   </p>
   <p lang="pt-br">Texto de um parágrafo com atributo lang.</p>
   <p title="outro_titulo">Texto de um parágrafo com atributo title = "outro_
   titulo".</p>
...
```

 [arquivo-2.2.6g.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor de atributo com filtros `p[title &= "lo"] [id]` para acessar os elementos parágrafo que contêm o atributo `title` com valor terminado na string `"lo"` e também um atributo `id` com qualquer valor aos quais será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

2.2.7 Filtros para seletores-filho

As pseudoclasses para seletores-filho são previstas nas CSS 3.

seletor:first-child

Acessa o elemento que é o primeiro filho do elemento definido em *`seletor`*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('ol li:first-child').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <ol>
        <li>Item de lista 1</li>
        <li>Item de lista 2</li>
        <li>Item de lista 3</li>
        <li>Item de lista 4</li>
        <li>Item de lista 5</li>
    </ol>
...
```

[arquivo-2.2.7a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `ol li:first-child` para acessar o elemento `li`, primeiro filho da lista ordenada `ol`, ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor:last-child

Acessa o elemento que é o último filho do elemento definido em *seletor*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('ol li:last-child').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <ol>
        <li>Item de lista 1</li>
        <li>Item de lista 2</li>
        <li>Item de lista 3</li>
        <li>Item de lista 4</li>
        <li>Item de lista 5</li>
    </ol>
...
```

[arquivo-2.2.7b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `ol li:last-child` para acessar o elemento `li`, último filho da lista ordenada `ol`, ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor:only-child

Acessa o elemento que é o filho único do elemento definido em *seletor*.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
```

```
            $('ol li:only-child').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <ol>
        <li>Item de lista 1</li>
        <li>Item de lista 2</li>
        <li>Item de lista 3</li>
        <li>Item de lista 4</li>
        <li>Item de lista 5</li>
    </ol>
    <ol>
        <li>Item único de lista</li>
    </ol>
...
```

[arquivo-2.2.7c.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `ol li:only-child` para acessar o elemento `li`, filho único que ocorre na segunda lista ordenada `ol`, ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

seletor:nth-child(*índice*/even/odd/*equação*)

Acessa o elemento que é filho do elemento definido em *seletor* e ocupa uma posição definida conforme mostrado a seguir.

Argumento	Descrição
índice	Um número inteiro começando com um para acessar o primeiro, segundo etc. filho. Exemplo: `nth-child(3)`.
even	Acessa os filhos de ordem par, lembrando que a contagem começa em 0 (zero) (par por padrão) e assim são pares os filhos na primeira, terceira etc. posição. Sintaxe: `nth-child(even)`.
odd	Acessa os filhos de ordem ímpar, lembrando que a contagem começa em 0 (zero) (par por padrão) e assim são ímpares os filhos na segunda, quarta etc. posição. Sintaxe: `nth-child(odd)`.
equação	Uma expressão matemática para acessar determinada posição. Exemplo: `nth-child(3n)` acessa os filhos nas posições 3, 6, 9, 12 etc. (n é a sequência de números naturais). Outro exemplo: `nth-child(3n - 2)` acessa os filhos nas posições 1, 4, 7, 10 etc.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('ol li:nth-child(3)').css('background', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Vermelha</button>
    <ol>
        <li>Item de lista 1</li>
        <li>Item de lista 2</li>
        <li>Item de lista 3</li>
        <li>Item de lista 4</li>
        <li>Item de lista 5</li>
    </ol>
...
```

[arquivo-2.2.7d.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `ol li:nth-child(3)` para acessar o elemento `li`, que é o terceiro filho na lista ordenada `ol`, ao qual será anexado o comportamento definido no script (mudar a cor de fundo para vermelha).

2.2.8 Seletores para formulários

:input

Acessa os elementos `input`, `textarea`, `select` e `button` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $(':input').css('border', '2px solid #ff0000');
        });
    });
</script>
</head>
```

```
<body>
   <button type="button">Bordas nos inputs</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input oculto: <input type="hidden" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input radio: <input type="radio" /></label></p>
   <p><label>input imagem: <input type="image" src="ok.gif" /></label></p>
   <p><label>Select: <select>
   <option>Opção 1</option>
   <option selected="selected">Opção selecionada</option>
   <option>Opção 3</option>
   <option disabled="disabled">Opção desabilitada</option>
   <option>Opção 4</option>
   <option>Opção 5</option>
   </select></label></p>
   <p><label>Textarea: <textarea cols="25" rows="8"></textarea></label></p>
   <p><label>input submit: <input type="submit" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input botao: <input type="button" /></label></p>
   <p><label>elemento botao: <button>Botão</button></label></p>
</form>
...
```

 [arquivo-2.2.8a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e usamos o seletor `:input` para acessar os elementos `input`, `textarea`, `select` e `button` do formulário aos quais será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:text

Acessa os elementos `input` do tipo `text` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':text').css('border', '2px solid #ff0000');
      });
   });
</script>
</head>
```

```
<body>
   <button type="button">Bordas nos inputs de texto</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input submit: <input type="submit" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input botao: <input type="button" /></label></p>
</form>
...
```

[arquivo-2.2.8b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:text` para acessar o elemento `input` do tipo `text` no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:password

Acessa os elementos `input` do tipo `password` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':password').css('border', '2px solid #ff0000');
      });
   });
</script>
</head>
<body>
   <button type="button">Bordas nos inputs de senha</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input submit: <input type="submit" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input botao: <input type="button" /></label></p>
</form>
...
```

[arquivo-2.2.8c.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:password` para acessar o elemento `input` do tipo `password` no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:radio

Acessa os elementos `input` do tipo `radio` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':radio').parent().css('border', '2px solid #ff0000');
      });
   });
</script>
</head>
<body>
   <button type="button">Bordas nos inputs radio</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input radio: <input type="radio" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input botao: <input type="button" /></label></p>
</form>
...
```

 [arquivo-2.2.8d.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:radio` para acessar o elemento `input` do tipo `radio` no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

> Neste script e no script a seguir, passou-se a estilização da borda para o elemento-pai `parent()` porque alguns navegadores, como Firefox 3, não suportam bordas em inputs dos tipos radio e checkbox.

:checkbox

Acessa os elementos input do tipo checkbox em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':checkbox').parent().css('border', '2px solid #ff0000');
      });
   });
</script>
</head>
<body>
   <button type="button">Bordas nos inputs checkbox</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input radio: <input type="radio" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input botao: <input type="button" /></label></p>
</form>
...
```

[arquivo-2.2.8e.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor pai do checkbox existente no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:submit

Acessa os elementos input do tipo submit em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':submit').css('border', '2px solid #ff0000');
      });
   });
</script>
```

```
</head>
<body>
   <button type="button">Bordas nos inputs submit</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input radio: <input type="radio" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input submit: <input type="submit" /></label></p>
   <p><label>input botao: <input type="button" /></label></p>
</form>
...
```

[arquivo-2.2.8f.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:submit` para acessar o elemento `input` do tipo `submit` no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:reset

Acessa os elementos `input` do tipo `reset` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':reset').css('border', '2px solid #ff0000');
      });
   });
</script>
</head>
<body>
   <button type="button">Bordas nos inputs reset</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input radio: <input type="radio" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input submit: <input type="submit" /></label></p>
```

```
    <p><label>input botao: <input type="button" /></label></p>
</form>
...
```

[arquivo-2.2.8g.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:reset` para acessar o elemento `input` do tipo `reset` no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:image

Acessa os elementos `input` do tipo `image` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $(':image').css('border', '2px solid #ff0000');
        });
    });
</script>
</head>
<body>
    <button type="button">Bordas nos inputs imagem</button>
<form action="">
    <p><label>input de texto: <input type="text" /></label></p>
    <p><label>input de senha: <input type="password" /></label></p>
    <p><label>input checkbox: <input type="checkbox" /></label></p>
    <p><label>input radio: <input type="radio" /></label></p>
    <p><label>input reset: <input type="reset" /></label></p>
    <p><label>input image: <input type="image" src="ok.gif" /></label></p>
    <p><label>input botao: <input type="button" /></label></p>
</form>
...
```

[arquivo-2.2.8h.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:image` para acessar o elemento `input` do tipo `image` no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:button

Acessa os elementos `input` do tipo `button` e os elementos `button` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':button').css('border', '2px solid #ff0000');
      });
   });
</script>
</head>
<body>
   <button type="button">Bordas nos inputs button</button>
<form action="">
   <p><label>input de texto: <input type="text" /></label></p>
   <p><label>input de senha: <input type="password" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input radio: <input type="radio" /></label></p>
   <p><label>input reset: <input type="reset" /></label></p>
   <p><label>input submit: <input type="submit" /></label></p>
   <p><label>input botao: <input type="button" /></label></p>
   <p><label>elemento button: <button> Botão</button></label></p>
</form>
...
```

[arquivo-2.2.8i.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e usamos o seletor `:button` para acessar o elemento `input` do tipo `button`, bem como o elemento `button` no formulário aos quais será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:file

Acessa os elementos `input` do tipo `file` em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
```

```
        $('button').click(function() {
            $(':file').css('boder', '2px solid #ff0000');
        });
    });
</script>
</head>
<body>
    <button type="button">Bordas nos inputs file</button>
<form action="">
    <p><label>input de texto: <input type="text" /></label></p>
    <p><label>input de senha: <input type="password" /></label></p>
    <p><label>input file: <input type="file" /></label></p>
    <p><label>input radio: <input type="radio" /></label></p>
    <p><label>input reset: <input type="reset" /></label></p>
    <p><label>input submit: <input type="submit" /></label></p>
    <p><label>input botao: <input type="button" /></label></p>
    <p><label>elemento button: <button> Botão</button></label></p>
</form>
...
```

[arquivo-2.2.8j.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor `:file` para acessar o elemento `input` do tipo `file` no formulário ao qual será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

Neste script, passou-se a estilização da borda para o elemento-pai `parent()` do input tipo `file` porque alguns navegadores, como Firefox 3, não suportam bordas em inputs do tipo `file`.

:hidden

Acessa os elementos `input` do tipo `hidden` em um formulário.

Veja [C2 S2.2.5] – Seletores de visibilidade.

Para limitar o acesso de qualquer um dos seletores de formulário estudados anteriormente a um determinado formulário em uma página, use um seletor mais específico. Por exemplo: `form#um input:text` limita o acesso ao campos de texto do formulário com `id="um"`.

2.2.9 Filtros para formulários

:enabled

Acessa os elementos que estejam habilitados em um formulário.

Em HTML, todos os elementos de um formulário estão habilitados por padrão. Você pode desabilitar elementos com o uso do atributo `disabled`.

:disabled

Acessa os elementos que estão desabilitados em um formulário.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $(':disabled').css('border', '2px solid #ff0000');
        });
    });
</script>
</head>
<body>
    <button type="button">Bordas nos campos disabled</button>
<form action="">
    <p><label>input de texto: <input type="text" disabled="disabled" /></label></p>
    <p><label>input checkbox: <input type="checkbox" /></label></p>
    <p><label>input checkbox: <input type="checkbox" /></label></p>
    <p><label>input checkbox: <input type="checkbox" checked="checked" /></label></p>
    <p><label>input checkbox: <input type="checkbox" /></label></p>
    <p><label>input radio: <input type="radio" /></label></p>
    <p><label>input radio: <input type="radio" /></label></p>
    <p><label>Select: <select>
    <option>Opção 1</option>
    <option selected="selected">Opção selecionada</option>
    <option>Opção 3</option>
    <option disabled="disabled">Opção desabilitada</option>
    <option>Opção 4</option>
    <option>Opção 5</option>
    </select></label></p>
    <p><label>input botao: <input type="button" disabled="disabled" /></label></p>
</form>...
```

 [arquivo-2.2.9a.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor :disabled para acessar todos os elementos no formulário aos quais será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

:checked

Acessa os elementos radio e checkbox em um formulário para os quais se tenha declarado o atributo checked.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $(':checked').parent().css('border', '2px solid #ff0000');
      });
   });
</script>
</head>
<body>
   <button type="button">Bordas nos campos checked</button>
<form action="">
   <p><label>input de texto: <input type="text" disabled="disabled" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input checkbox: <input type="checkbox" checked="checked" /></label></p>
   <p><label>input checkbox: <input type="checkbox" /></label></p>
   <p><label>input radio: <input type="radio" /></label></p>
   <p><label>input radio: <input type="radio" checked="checked" /></label></p>
</form>
...
```

[arquivo-2.2.9b.html]

Nesse exemplo, utilizaram-se um botão para disparar o evento e o seletor :checked para acessar todos os elementos no formulário que tenham o atributo checked aos quais será anexado o comportamento definido no script (adicionar uma borda na cor vermelha).

Neste script, passou-se a estilização da borda para o elemento-pai parent() dos elementos com atributo checked porque alguns navegadores, como Firefox 3, não suportam bordas em elementos com atributo checked.

:selected

Acessa os valores que tenham sido selecionados em controles de um formulário.

Exemplo:

```
...
<style type="text/css" media="all">
   b{color:#c30;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('select').change(function() {
         $('span').empty();
         $('option:selected').each(function() {
            $('span').append('Você selecionou: <b>' + $(this).text() + '</b><br />');
         });
      });
   });
</script>
</head>
<body>
<form action="" method="">
   <p><label>Selecione um estado:<br /><select multiple="multiple">
   <option>Amazonas</option>
   <option>Acre</option>
   <option>Ceará</option>
   <option>Espirito Santo</option>
   <option>Goiás</option>
   </select></label></p>
</form>
<span></span>
...
```

 [arquivo-2.2.9c.html]

Para usar um dos metacaracteres empregado na sintaxe dos seletores do tipo (`:[ ]`) em uma string como parte de um nome, escape o caractere com duas \\ (barras invertidas). Por exemplo: para acessar um elemento com o seletor de atributo `name`, se o valor de `name` é `xp]to`, ao definir o seletor, use `$('[name = "xp\\[to"]')` em lugar de `$('[name = "xp[to"]')`. Melhor ainda: evite usar esses metacaracteres em sua marcação.

CAPÍTULO 3

Métodos de manipulação do DOM

Neste capítulo, serão estudadas as funcionalidades disponíveis em jQuery para inspecionar e definir atributos e seus respectivos valores aos elementos componentes de uma página.

3.1 Manipulação de atributos gerais

seletorjQuery.attr(*nome_atributo*)

Acessa o valor do atributo definido no parâmetro `nome_atributo` para o primeiro elemento encontrado pelo `seletorjQuery`. Caso não exista o atributo na marcação, o valor retornado será `undefined` (indefinido).

Exemplo:

```
...
<style type="text/css" media="all">
#resultado {display:none; width:300px; padding:8px 12px; border:1px solid #000;
   background:#ff0;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
      1. $(document).ready(function() {
      2.     $('button').click(function() {
      3.         var valorAtributo = $('h1').attr('title');
      4.         $('#resultado').css('display', 'block')
                 .text('O valor do atributo title é: ' + valorAtributo);
      5.     });
      6. });
```

```
</script>
</head>
<body>
   <button type="button">Atributo</button>
   <h1 title="logotipo">Cabeçalho nível 1 com atributo title="logotipo"</h1>
   <p id="um">Primeiro parágrafo com atributo id="um"</p>
   <p class="diferente">segundo parágrafo com atributo class="diferente"</p>
   <p class="diferente">segundo parágrafo com atributo class="diferente"</p>

   <div id="resultado"></div>
...
```

[arquivo-3.1a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Acompanhe a seguir o funcionamento de cada linha do código desenvolvido para este exemplo.

Código comentado:

Linha(s)	Descrição
Linhas 1 e 6	Container obrigatório para todo script jQuery. Veja [C1 S1.1.6.2].
Linha 2	Define uma função a ser executada quando o usuário clica o botão existente na página.
Linha 3	Pega o valor do atributo `title` do primeiro elemento `h1` que aparece na marcação e armazena na variável `valorAtributo`.
Linha 4	Seleciona o `div#resultado` que foi inicialmente escondida com a regra CSS `#resultado {display:none;}` e a torna visível com o método jQuery `css('display', 'block')`. Em seguida, usando o método jQuery `text('O valor do atributo title é: ' + valorAtributo)`, escreve dentro do `div#resultado` a frase apresentando o valor do atributo acessado, que havia sido armazenado em `valorAtributo`.

Veja o funcionamento desse script clicando o botão "Atributo" no arquivo indicado, disponível no site do livro.

seletorjQuery.attr({*atributo*:*valor*})

Acessa o elemento definido em *`seletorjQuery`* e insere o par *`atributo = "valor"`* no elemento. Você pode inserir quantos pares quiser. No exemplo a seguir, inseriram-se dois pares.

Exemplo:

```
...
<style type="text/css" media="all">
   img {display:none;}
</style>
```

```
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
      1. $(document).ready(function() {
      2.     $('button').click(function() {
      3.         $('img').css('display', 'block').attr({
      4.             src:"maujor.jpg",
      5.             alt:"Foto do Maujor"
      6.         });
      7.     });
      8. });
</script>
</head>
<body>
   <img />
...
```

[arquivo-3.1b.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Acompanhe a seguir o funcionamento de cada linha do código desenvolvido para esse exemplo.

Note que se inseriu um elemento `img` vazio no documento e escondeu-se com a declaração CSS `img {display:none;}`.

Código comentado:

Linha(s)	Descrição
Linhas 1 e 7	Container obrigatório para todo script jQuery. Veja [C1 S1.1.6.2].
Linha 2	Define uma função a ser executada quando o usuário clica o botão existente na página.
Linha 3	Seleciona o(s) elemento(s) `img` existente(s) na página (no exemplo, há um elemento) e revela-o com uma declaração CSS. A seguir, insere no elemento os atributos `src` que definem o caminho para uma imagem e o atributo `alt` que descreve sumariamente a imagem.

Veja o funcionamento desse script clicando o botão "Atributo" no arquivo indicado, disponível no site do livro.

seletorjQuery.attr(*atributo*, *valor*)

Acessa o elemento definido em *`seletorjQuery`* e insere o par *`atributo = "valor"`* no elemento. Ao contrário do método anterior, este método permite inserir somente um par atributo/valor.

Exemplo:

```
...
<style type="text/css" media="all">
   img {display:none;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('img').css('display', 'block').attr('src', 'maujor.jpg');
      });
   });
</script>
</head>
<body>
   <img />
...
```

[arquivo-3.1c.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. O funcionamento deste script é idêntico ao anterior, mas aqui não se inseriu o atributo `alt`.

Veja o funcionamento desse script clicando o botão "Atributo" no arquivo indicado, disponível no site do livro.

seletorjQuery.**removeAttr**(*atributo*)

Acessa o elemento definido em *`seletorjQuery`* e remove o `atributo` definido no parâmetro *`atributo`*.

Exemplo:

```
...
<style type="text/css" media="all">
   div {width:300px; height:175px; border:1px solid #000;}
   #remover {background:#0f0;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('#remover').removeAttr('id');
      });
   });
</script>
</head>
```

```
<body>
    <button type="button">Remover</button>
    <div id="remover"></div>
...
```

 [arquivo-3.1d.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que um `div` com `id = "remover"` é estilizado com CSS. A cor de fundo verde (`#0f0`) foi declarada em uma regra CSS para o `div`. Se remover o `id` do `div`, a regra CSS perderá o efeito. O script faz exatamente isso, remove o `id` e o `div` perde o fundo verde.

Veja o funcionamento desse script clicando o botão "Remover" no arquivo indicado, disponível no site do livro.

3.2 Manipulação do atributo `class`

***seletorjQuery*.addClass(*valor_classe*)**

Acessa o elemento definido em *`seletorjQuery`* e atribui-lhe uma classe de nome igual a *`valor_classe`*.

Segundo as especificações para a HTML, o atributo `class` aceita um ou mais valores. No caso de mais de um valor, os nomes devem ser separados por espaço. Exemplo de parágrafo com três classes diferentes: `<p class = "um sete outra">`.

Exemplo:

```
...
<style type="text/css" media="all">
    p.minha_classe {color:#f00; font-size:45px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('p').addClass('minha_classe');
        });
    });
</script>
</head>
<body>
    <button type="button">Adicionar</button>
    <p>Parágrafo ao qual foi adicionada uma classe com jQuery.</p>
...
```

 [arquivo-3.2a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que um parágrafo `p`, sem atributos, não foi estilizado. Uma regra CSS para parágrafos com a classe `minha_classe` define cor de texto vermelha e tamanho de fonte 45px. Com o uso de jQuery, insere-se a classe `minha_classe` no parágrafo e este assume a estilização constante da regra CSS.

Veja o funcionamento desse script clicando o botão "Adicionar" no arquivo indicado, disponível no site do livro.

seletorjQuery.**hasClass**(*valor_classe*)

Acessa o elemento definido em *seletorjQuery* e verifica se a classe de valor igual a *valor_classe* existe para tal elemento no documento. Retorna o valor booleano TRUE ou FALSE.

seletorjQuery.**removeClass**(*valor_classe*)

Acessa o elemento definido em *seletorjQuery* e remove a classe cujo valor foi definido no parâmetro *valor_classe*.

Exemplo:

```
...
<style type="text/css" media="all">
   p.remover {background:#ff0; border:1px solid #000;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').removeClass('remover');
      });
   });
</script>
</head>
<body>
   <button type="button">Remover</button>
   <p class="remover">Parágrafo com a classe="remover" definida na marcação.</p>
...
```

[arquivo-3.2b.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que um parágrafo `p` com `class = "remover"` é estilizado com CSS, que lhe adiciona uma cor de fundo amarela e uma borda na cor preta. Se remover essa classe, a regra CSS perderá o efeito. O script faz exatamente isso, remove o atributo classe do parágrafo e este perde o fundo amarelo e a borda definida pela CSS.

Veja o funcionamento desse script clicando o botão "Remover" no arquivo indicado, disponível no site do livro.

seletorjQuery.**toggleClass**(*valor_classe*)

Acessa o elemento definido em *seletorjQuery* e remove a classe cujo nome foi definido no parâmetro *valor_classe* se este estiver presente ou adiciona o valor caso não esteja presente.

Exemplo:

```
...
<style type="text/css" media="all">
   p.remover {background:#ff0; border:1px solid #000;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').toggleClass('remover');
      });
   });
</script>
</head>
<body>
   <button type="button">Mudar</button>
   <p class="remover">Parágrafo com a classe="remover" definida na marcação.</p>
...
```

[arquivo-3.2c.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Aqui cada vez que você clica o botão, a classe do elemento `p` é alterada e o script entra em ação. Note que com a classe, o parágrafo recebe uma estilização definida nas regras de estilo.

Veja o funcionamento desse script clicando seguidamente o botão "Mudar" no arquivo indicado, disponível no site do livro.

3.3 Manipulação de conteúdos html

seletorjQuery.html()

Acessa o conteúdo `html` (conteúdo e marcação) dentro do elemento definido em *`seletorjQuery`*.

Exemplo:

```
...
<style type="text/css" media="all">
   span {display:none; background:#ff0; padding:8px 12px; border:1px solid #000;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var conteudoHtml = $('p').html();
         $('span').css('display', 'block').text(conteudoHtml);
      });
   });
</script>
</head>
<body>
   <button type="button">HTML</button>
   <p>Esse é o 1<sup>o.</sup> parágrafo para <em>demonstrar</em> o método
   <strong>jQuery</strong><code>html()</code>.</p>
   <p>Outro parágrafo.</p>
   <span></span>
...
```

[arquivo-3.3a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se usou o método jQuery `html()` para selecionar a marcação HTML do primeiro parágrafo `p` que aparece na marcação. O resultado foi armazenado na variável `conteudoHtml` e a seguir se escreveu o resultado em um elemento `span` que havia sido oculto por regras CSS.

Veja o funcionamento desse script clicando o botão "HTML" no arquivo indicado, disponível no site do livro.

Note que o elemento `span` é vazio. Tivemos que ocultá-lo com regra CSS porque ele foi estilizado com uma borda e um fundo, fazendo com que estas fossem renderizadas, mesmo estando o elemento vazio.

seletorjQuery.**html(*valor*)**

Acessa o elemento definido em `seletorjQuery` e nele insere o `html` (conteúdo e marcação) definido no parâmetro *`valor`*.

Exemplo:

```
...
<style type="text/css" media="all">
   p.um {color:#f00; font:24px sans-serif;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').html('<p class="um">Conteúdo inserido com <code>html(<i>valor</i>)
            </code></p>');
      });
   });
</script>
</head>
<body>
   <button type="button">HTML</button>
   <div></div>
...
```

[arquivo-3.3b.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se usou o método jQuery `html(valor)` para inserir um parágrafo `p` dentro de um `div`. O parágrafo recebeu uma classe que havia sido estilizada por regras CSS.

Veja o funcionamento desse script clicando o botão “HTML” no arquivo indicado, disponível no site do livro.

3.4 Manipulação de textos

seletorjQuery.**text()**

Acessa o conteúdo textual do elemento definido em `seletorjQuery`.

Exemplo:

```
...
<style type="text/css" media="all">
   span {display:none; background:#ff0; padding:8px 12px; border:1px solid #000;}
</style>
```

```
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var conteudoTextual = $('p').text();
         $('span').css('display', 'block').text(conteudoTextual);
      });
   });
</script>
</head>
<body>
   <button type="button">HTML</button>
   <p>Esse é o 1<sup>o.</sup> parágrafo para <em>demonstrar</em> o método
   <strong>jQuery</strong><code>text()</code>.</p>
   <span></span>
...
```

[arquivo-3.4a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se usou o método jQuery `text()` para inspecionar o conteúdo `html` do primeiro parágrafo `p` que aparece na marcação. O resultado foi armazenado na variável `conteudoTextual` e a seguir se escreveu o resultado em um elemento `span` que havia sido oculto por regras CSS.

Veja o funcionamento desse script clicando o botão "Texto" no arquivo indicado, disponível no site do livro.

seletorjQuery.text(*valor*)

Acessa o elemento definido em *`seletorjQuery`* e nele insere o conteúdo definido no parâmetro *`valor`*.

Exemplo:

```
...
<style type="text/css" media="all">
   div {color:#f00; font:24px sans-serif;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').text('<p class="um">Conteúdo inserido com <code>text(<i>val</i>)
            </code></p>');
      });
   });
```

```
</script>
</head>
<body>
    <button type="button">Texto</button>
    <div></div>
...
```

[arquivo-3.4b.html]

Nesse exemplo, utiliza-se um botão para disparar o evento. Observe que se usou o método jQuery `text(valor)` para inserir um parágrafo `p` dentro de um `div`. O parágrafo recebeu uma classe, um texto marcado com o elemento `code` e outro com o elemento `i`, mas tudo foi ignorado e inserido como string, pois o método jQuery destina-se a inserir texto puro e não insere marcação. Para inserir marcação, use `html(valor)` como visto anteriormente.

Veja o funcionamento desse script clicando o botão "Texto" no arquivo indicado, disponível no site do livro.

3.5 Manipulação de valores

seletorjQuery.val()

Acessa o valor do atributo `value` do elemento definido em *`seletorjQuery`*.

Exemplo:

```
...
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            var valorValue = $('input').val();
            $('span').text(valorValue);
        });
    });
</script>
</head>
<body>
    <button type="button">Valor</button><br />
    <input type="text" value="Entre um texto qualquer" /><br />
    <span></span>
...
```

[arquivo-3.5a.html]

No exemplo, utilizou-se um botão para disparar o evento. Observe que se usou o método jQuery `val()` para inspecionar o valor do atributo `value` do campo de texto. Definiu-se na marcação HTML um valor inicial para o atributo. Se o usuário entrar com um valor diferente do inicial, o script reterá tal valor. O script armazena o valor na variável `valorValue` e, a seguir, escreve o resultado em um elemento `span`.

Veja o funcionamento desse script, no arquivo indicado, disponível no site do livro, inicialmente clicando o botão "Valor" existente na página. A seguir, preencha qualquer texto no campo e clique o botão "Valor" novamente.

seletorjQuery.val(*valor_atributo_value*)

Insere, no elemento definido em *`seletorjQuery`*, o atributo `value` com o valor definido pelo parâmetro *`valor_atributo_value`*.

Exemplo:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('input').val('Valor inserido com jQuery');
      });
   });
</script>
</head>
<body>

   <button type="button">Valor</button><br />
   <input type="text" size="30" />
...
```

[arquivo-3.5b.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se usou o método jQuery `val(valor)` para inserir o texto definido em `valor` no campo `input`.

Veja o funcionamento desse script clicando o botão "Valor" no arquivo indicado, disponível no site do livro.

3.6 Manipulação de conteúdos

seletorjQuery.**append**(*conteúdo*)

Insere o conteúdo definido no parâmetro `conteúdo` logo após o conteúdo do elemento definido em `seletorjQuery`. O parâmetro `conteúdo` pode ser tanto marcação HTML como texto puro.

Exemplo:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').append(' <strong>Aqui texto inserido com jQuery</strong>');
      });
   });
</script>
</head>
<body>

   <button type="button">Inserir texto</button><br />
   <p>Texto dentro do elemento parágrafo.</p>
...
```

[arquivo-3.6a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se usou o método jQuery `append(conteúdo)` para inserir marcação html e texto definidos no parâmetro `conteúdo`, logo após o conteúdo existente no elemento parágrafo.

Veja o funcionamento desse script clicando o botão "Inserir com append()" no arquivo indicado, disponível no site do livro.

O parâmetro `conteúdo` desse método pode ser marcação já existente na árvore do documento e, neste caso, duas situações são possíveis:

- **Primeira situação**: a inserção ocorrerá em um só alvo, ou seja, apenas um elemento receberá o conteúdo a ser inserido. O exemplo a seguir mostra tal situação, na qual um link já existente na marcação será inserido dentro de um parágrafo.

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
```

```
            $('p').append($('a'));
        });
    });
</script>
</head>
<body>
    <button type="button">Inserir texto</button><br /><br />
    <a href="#">Aqui um link</p>
    <p>Texto dentro do elemento parágrafo. </p>
...
```

[arquivo-3.6b.html]

Como você poderá constatar no arquivo indicado, o script retira o link de sua posição no documento e insere-o após o conteúdo existente no parágrafo, ou seja, o conteúdo a inserir foi movido de sua posição para a posição definida em `append()`.

- **Segunda situação**: a inserção ocorrerá em mais de um alvo, ou seja, dois ou mais elementos receberão o conteúdo a ser inserido. O exemplo a seguir mostra tal situação, na qual um link já existente na marcação será inserido dentro de três parágrafos.

```
...
<script type="text/javascr"pt">
    $(document).ready(function() {
        '$('button').click(function() {
            '$'('p').append($('a'));
        });
    });
</script>
</head>
<body>
    <button type="button">Inserir texto</button><br /><br />
    <a href="#">Aqui um link</p>
    <p>Texto dentro do elemento parágrafo. </p>
    <p>Texto dentro do elemento parágrafo. </p>
    <p>Texto dentro do elemento parágrafo. </p>
...
```

[arquivo-3.6c.html]

Como você poderá constatar no arquivo indicado, o script faz cópias do link e insere-as após o conteúdo de cada um dos três parágrafos, ou seja, o conteúdo a inserir permanece em sua posição inicial, sendo copiado para as posições definidas em `append()`.

Mover e copiar

Como se mostrou nos dois arquivos anteriores, o método jQuery `append()`, quando toma como parâmetro marcação previamente existente no documento, comporta-se de maneiras distintas. Se o alvo de `append()` é único no documento, a marcação tomada como parâmetro é movida de seu lugar no documento para a posição onde será inserida. Se o alvo de `append()` é múltiplo, são feitas cópias da marcação para as inserções e a marcação original permanece em seu lugar.

seletorjQuery.appendTo(*alvo*)

Insere o conteúdo definido no *`seletorjQuery`* logo após o conteúdo do elemento definido no parâmetro *`alvo`*. Este método cumpre as mesmas finalidades do método `append()` estudado no item anterior, usando uma sintaxe variante à sintaxe desse método.

Exemplo:

```
...
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('<strong>Aqui texto inserido com jQuery</strong>').appendTo('p');
        });
    });
</script>
</head>
<body>
    <button type="button">Inserir com appendTo()</button><br />
    <p>Texto dentro do elemento parágrafo.</p>
...
```

 [arquivo-3.6d.html]

Tal como estudado para o método append(), este método também admite inserção de conteúdos já existentes na árvore do documento e as duas situações estudadas para aquele método são válidas para este, ou seja, havendo um alvo, o elemento é movido; havendo dois ou mais alvos, o conteúdo movido é cópia do original.

seletorjQuery.**prepend(***conteúdo***)**

seletorjQuery.**prependTo(***alvo***)**

Estes dois métodos são equivalentes aos métodos `append()` e `appendTo()`, com a diferença de que a inserção do conteúdo se faz antes do conteúdo que receberá a inserção.

seletorjQuery.**after(***conteúdo***)**

seletorjQuery.**insertAfter(***alvo***)**

Estes dois métodos são equivalentes aos métodos `append()` e `appendTo()`, com a diferença de que a inserção do conteúdo se faz antes do elemento (e não do conteúdo do elemento) que receberá a inserção. Veja o mesmo exemplo mostrado para `append()` – `arquivo-3.6a.html`) – alterando o método para `after()`, com a finalidade de caracterizar a diferença. Compare o arquivo desse exemplo com o `arquivo-3.6a.html`.

Exemplo:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').after('<strong>Aqui texto inserido com jQuery</strong>');
      });
   });
</script>
</head>
<body>
   <button type="button">Inserir com after()</button><br />
   <p>Texto dentro do elemento parágrafo.</p>
...
```

[arquivo-3.6e.html]

Conforme se pode constatar nos dois arquivos (`arquivo-3.6a.html` e `arquivo-3.6e.html`), o método `append()` coloca a marcação do elemento `span` em linha com a do parágrafo, ficando claro que foi inserida dentro do parágrafo, e o método `after()` causa renderização da marcação `span` embaixo, indicando que foi inserida fora do parágrafo.

Para esses dois métodos, são válidos os critérios de mover e copiar, como descritos para os métodos `append()` e `appendTo()`.

seletorjQuery.**before(***conteúdo***)**

seletorjQuery.**insertBefore(***alvo***)**

Estes dois métodos são equivalentes aos métodos anteriores, com a diferença de que o conteúdo é inserido antes do alvo.

Para esses dois métodos, são válidos os critérios de mover e copiar, como descritos para os métodos `append()` e `appendTo()`.

seletorjQuery.**wrap(***html***)**

Cria o container definido no parâmetro *`html`* para cada um dos elementos definidos em *`seletorjQuery`*.

Exemplo:

```
...
.container {border:1px solid #000; background:#ffc; margin:20px 0;}
</style>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').wrap('<div class="container"></div>');
      });
   });
</script>
</head>
<body>
   <button type="button">Definir container</button><br />
   <p>Texto dentro do elemento parágrafo.</p>
   <p>Texto dentro do elemento parágrafo.</p>
   <p>Texto dentro do elemento parágrafo.</p>
...
```

[arquivo-3.6f.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que a marcação passada pelo parâmetro do método é um `div` com `class = "container"` que será inserida na marcação como um container para cada um dos parágrafos. Para facilitar a visualização do funcionamento do script, definiu-se uma regra CSS para a classe `.container`. Veja o funcionamento desse script clicando o botão "Definir container" no arquivo indicado, disponível no site do livro.

seletorjQuery.**wrap(***elemento***)**

O parâmetro `elemento` é um container para os elementos definidos em `seletorjQuery`, mas em vez de ser escrito no método, é escrito com JavaScript.

O container definido deve ser um elemento vazio, isto é, não pode haver conteúdos dentro dele.

Exemplo:

```
...
div {border:1px solid #000; background:#ffc; margin:20px 0;}
</style>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').wrap(document.createElement('div'));
      });
   });
</script>
</head>
<body>
   <button type="button">Definir container</button><br />
   <p>Texto dentro do elemento parágrafo.</p>
...
```

[arquivo-3.6g.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que o container é um `div` criado com o método `document.createElement('div')`. Para facilitar a visualização do funcionamento do script, definiu-se uma regra CSS para o `div`. Veja o funcionamento desse script clicando o botão "Definir container" no arquivo indicado, disponível no site do livro.

seletorjQuery.**wrapAll(***html***)**

Cria o container definido no parâmetro `html`, para todos os elementos definidos em `seletorjQuery`. A diferença para `wrap(html)` é que naquele método o container é para cada um dos elementos e neste, para todos.

Exemplo:

```
...
.container {border:1px solid #000; background:#ffc; margin:20px 0;}
</style>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').wrapAll('<div class="container"></div>');
      });
   });
</script>
</head>
<body>
   <button type="button">Definir container</button><br />
   <p>Texto dentro do elemento parágrafo.</p>
   <p>Texto dentro do elemento parágrafo.</p>
   <p>Texto dentro do elemento parágrafo.</p>
...
```

[arquivo-3.6h.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que a marcação passada pelo parâmetro do método é um `div` com `class = "container"` que será inserido na marcação como um container para todos os parágrafos. Para facilitar a visualização do funcionamento do script, definiu-se uma regra CSS para a classe `.container`. Veja o funcionamento desse script clicando o botão "Definir container" no arquivo indicado, disponível no site do livro.

O uso desse método causa uma alteração substancial na árvore do documento. No exemplo mostrado, os três parágrafos estão em sequência. Considere um exemplo mais complexo com vários parágrafos em uma página, intercalados por outros elementos. Esse método cria o container e "puxa" todos os parágrafos para uma posição imediatamente após o primeiro parágrafo encontrado pelo seletor, colocando todos no container. Daí é fácil concluir a alteração no DOM.

seletorjQuery.wrapAll(*elemento*)

Este método é semelhante ao método `wrap(elemento)`, com a diferença de que o container é para todos os elementos-alvo.

seletorjQuery.**wrapInner**(*html*)

Cria o container definido no parâmetro `html` para os conteúdos (não para os elementos) de cada um dos elementos definidos em `seletorjQuery`.

seletorjQuery.**wrapInner**(*elemento*)

Apresenta o mesmo efeito do anterior, mas neste método o container é criado com JavaScript.

seletorjQuery.**remove**(*filtro*)

Este método remove todas as ocorrências do elemento definido em `seletorjQuery`. O parâmetro `filtro` é opcional e pode ser usado para filtrar ocorrências específicas da remoção.

Exemplo:

```
...
</style>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').remove('.remover');
      });
   });
</script>
</head>
<body>
   <button type="button">Remover</button><br />
   <p>Texto dentro do elemento parágrafo.</p>
   <p class="remover">Texto dentro do elemento parágrafo com classe remover.</p>
   <p>Texto dentro do elemento parágrafo.</p>
...
```

[arquivo-3.6i.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que o método remove todos os parágrafos do documento cujo atributo classe tem o valor `remover`. O método remove o elemento do DOM. Veja o funcionamento desse script clicando o botão “Remover” no arquivo indicado, disponível no site do livro.

seletorjQuery.empty()

Este método remove todos os conteúdos do elemento definido em `seletorjQuery`. Funciona como o método anterior, removendo conteúdos, sem remover o elemento. As tags de abertura e fechamento do elemento, permanecem no DOM.

Exemplo de sintaxe:

```
$('p').empty();
```

seletorjQuery.replaceWith(*conteúdo*)

Substitui o elemento definido no `seletorjQuery` por aquele definido no parâmetro `conteúdo`. Este parâmetro pode ser marcação HTML (como mostrado no exemplo) ou elemento e seus conteúdos criados com JavaScript.

Exemplo:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').replaceWith('<h1>Cabeçalho substitui parágrafo</h1>');
      });
   });
</script>
</head>
<body>
   <button type="button">Substituir</button><br />
   <p>Texto dentro do elemento parágrafo.</p>
...
```

[arquivo-3.6j.html]

Veja o funcionamento desse script clicando o botão “Substituir” no arquivo indicado, disponível no site do livro.

conteúdo.replaceAll(*seletorjQuery*)

Este método cumpre a mesma finalidade do anterior com uma sintaxe reversa, ou seja, substitui o seletor definido em `seletorjQuery` pelo parâmetro `conteúdo`, que pode ser marcação HTML ou elemento e conteúdos criados com JavaScript.

O exemplo a seguir faz o mesmo efeito daquele mostrado no exemplo do arquivo anterior (`arquivo-3.6j.html`).

```
$('<h1>Cabeçalho substitui parágrafo</h1>').replaceAll('p');
```

seletorjQuery.clone()

Cria uma cópia do *`seletorjQuery`*.

Exemplo:

```
...
<style type="text/css" media="all">
div {border:1px solid #000; background:#ffc;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').clone().appendTo('div');
      });
   });
</script>
</head>
<body>
   <button type="button">Clonar</button><br />
   <p>Parágrafo a clonar.</p>
   <div>div para receber o clone</div>...
```

[arquivo-3.6k.html]

Veja o funcionamento desse script clicando o botão "Clone" no arquivo indicado, disponível no site do livro.

seletorjQuery.clone(true)

Idêntico ao anterior, cria uma cópia do *`seletorjQuery`* e de todos os scripts a ele anexados.

CAPÍTULO 4

CSS e inspeção do DOM

Neste capítulo, serão estudadas as funcionalidades disponíveis em jQuery para definir estilização aos elementos componentes da árvore do documento. Também se abordarão os métodos para acessar e selecionar componentes do DOM.

Nos títulos de cada item usamos o termo seletorjQuery, genericamente, para indicar um seletor, um método de seleção ou a seleção resultante de um encadeamento.

4.1 Estilização geral

seletorjQuery.**css**(*propriedade*)

Acessa o valor definido para a propriedade CSS do elemento `seletorjQuery`. A propriedade a acessar deve ser declarada no parâmetro `propriedade`.

Exemplo:

```
...
<style type="text/css" media="all">
div {width:100px; height:100px; margin:20px 0; border:1px solid #000; background:#ffc;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var corFundo = $('div').css('backgroundColor'); // ver dica após o script
         $('span').text('A cor de fundo do div é: ' + corFundo);
      });
   });
</script>
```

```
</head>
<body>
    <button type="button">CSS</button><br />
    <div>DIV estilizado com CSS</div>
    <span></span>
...
```

 [arquivo-4.1a.html]

Propriedades CSS com nomes compostos de duas palavras cuja sintaxe CSS adota separação com hífen, tais como `background-color` e `line-height`, devem ser escritas em sintaxe jQuery com a notação conhecida como "camelCase", ou seja, `backgroundColor` e `lineHeight` seguindo a sintaxe formal JavaScript.

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se acessou o valor da propriedade CSS `background-color` para o `div` e armazenou-se esse valor em uma variável denominada *corFundo*. A seguir, escreveu-se o resultado em um elemento `span` existente na marcação. Veja o funcionamento desse script clicando o botão "CSS" no arquivo indicado, disponível no site do livro.

seletorjQuery.css('*propriedade*','*valor*')

Estiliza o elemento *seletorjQuery* com a declaração CSS definida nos parâmetros *propriedade*, *valor*. *Propriedade* é a propriedade CSS a estilizar e *valor*, seu valor ou característica.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('div').css('border', '1px solid red');
        });
    });
</script>
</head>
<body>

    <button type="button">Estilizar</button><br />
    <div>DIV estilizado com regra CSS declarada por jQuery</div>
...
```

 [arquivo-4.1b.html]

Veja o funcionamento desse script clicando o botão "Estilizar" no arquivo indicado, disponível no site do livro.

seletorjQuery.css({*propriedade*:'*valor*', *propriedade*:'*valor*', ...})

Estiliza o elemento `seletorjQuery` com as declarações CSS definidas no parâmetro `propriedade:valor`. A diferença para a anterior é que com este método se pode definir duas ou mais declarações CSS.

Exemplo:

```
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('div').css({
                width:'400px',
                lineHeight:'60px',        // camelCase para lineHeight
                background:'#ff0',
                border:'1px dotted #000
            });
        });
    });
</script>
</head>
<body>
    <button type="button">Estilizar</button><br />
    <div>DIV estilizado com regra CSS declarada por jQuery</div>
...
```

[arquivo-4.1c.html]

Veja o funcionamento desse script clicando o botão "Estilizar" no arquivo indicado, disponível no site do livro.

4.2 Posicionamento

seletorjQuery.offset()

Acessa o valor das coordenadas CSS `top` e `left` do elemento `seletorjQuery` em relação à viewport.

Exemplo:

```
...
<style type="text/css" media="all">
   div {width:100px; height:100px; background:#0f0; margin:50px 0 0 100px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var coordenadasDiv = $('div').offset();
         $('span').html('As coordenadas do div são:<br /> left: ' +
            coordenadasDiv.left + ' px<br /> top: ' + coordenadasDiv.top +' px');
      });
   });
</script>
</head>
<body>
   <button type="button">Coordenadas</button><br />
   <div>DIV para determinar as coordenadas</div>
   <span></span>
...
```

[arquivo-4.2a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se acessaram os valores das coordenadas, os quais se armazenaram em uma variável denominada `coordenadasDiv`. A seguir, escreveu-se o resultado em um elemento `span` existente na marcação.

Veja o funcionamento desse script clicando o botão "Coordenadas" no arquivo indicado, disponível no site do livro.

seletorjQuery.position()

Acessa o valor das coordenadas CSS `top` e `left` do elemento *`seletorjQuery`* em relação ao `offset` do elemento-pai.

Exemplo:

```
...
<style type="text/css" media="all">
   div {
      width:300px;
      height:200px;
      background:#0f0;
      margin:50px 0 0 100px;
      }
```

```
        p {background:red;}
    </style>
    <script type="text/javascript" src="../jquery-1.2.6.js"></script>
    <script type="text/javascript">
        $(document).ready(function() {
            $('button').click(function() {
                var coordenadasPara = $('p:eq(2)').position();
                $('span').html('As coordenadas do 3o. parágrafo são:<br /> left: ' +
                    coordenadasPara.left + ' px<br /> top: '+ coordenadasPara.top +' px');
            });
        });
    </script>
    </head>
    <body>
        <button type="button">Coordenadas</button><br />
        <div>DIV para determinar as coordenadas</div>
        <span></span>
    ...
```

[arquivo-4.2b.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se acessaram os valores das coordenadas do terceiro parágrafo, os quais se armazenaram em uma variável denominada `coordenadasPara`. A seguir, escreveu-se o resultado em um elemento `span` existente na marcação.

Veja o funcionamento desse script clicando o botão "Coordenadas" no arquivo indicado, disponível no site do livro.

> Coordenadas de posicionamento de um elemento em relação a seu ancestral dependem do contexto CSS de posicionamento. No exemplo desenvolvido, tanto o elemento-pai como o elemento-filho não foram posicionados com declaração CSS, assim o contexto de posicionamento para ambos é a viewport e as coordenadas a ela serão referenciadas.

Note que, ao executar o script do exemplo, a coordenada `left` para o parágrafo é igual à coordenada `left` para o `div` e a coordenada `top` é igual à do `div` mais um valor igual à distância do topo do parágrafo ao topo do `div`.

A melhor maneira de entender o funcionamento desse método é fazer algumas cópias do arquivo de exemplo e nelas alterar o posicionamento do `div` e do parágrafo com regras CSS (use regras CSS in-line para simplificar). Execute o script em cada situação criada para ver o resultado. Comece declarando `position: relative` para o `div` e `position:absolute; top:0; left:0` para o parágrafo.

***seletorjQuery*.scrollTop()**

Acessa o valor da coordenada `top` que mede a distância em pixel de uma rolagem vertical.

Exemplo:

```
...
<style type="text/css" media="all">
   div {
      width:400px;
      height:250px;
      border:1px dotted #000;
      margin:20px 0;
      padding:5px 10px;
      background:#d6e2e5;
      overflow:scroll;
      }
   p {width:600px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
1.    $(document).ready(function() {
2.       $('button:eq(0)').click(function() {
3.       var rolagemVertical = $('#rolagem').scrollTop();
4.       var rolagemHorizontal = $('#rolagem').scrollLeft();
5.       $('span').html('Posição da rolagem vertical: ' + rolagemVertical +
            ' px<br />Posição da rolagem horizontal: ' + rolagemHorizontal + ' px');
6.       });
7.       $('button:eq(1)').click(function() {
8.       $('#rolagem').scrollTop(0);
9.       $('#rolagem').scrollLeft(0);
10.      $('span').empty();
11.      });
12    });
</script>
</head>
<body>
<p>Role as barras para qualquer posição e clique o botão <strong>Coordenadas scroll
   </strong></p>
   <button type="button">Coordenadas scroll</button>   
   <button type="reset">Reset</button><br />
<div id="rolagem">
   <p>Lorem ipsum
   ..muito conteúdo..
</div>
<span></span>
</body>
</html
```

 [arquivo-4.2c.html]

Acompanhe a seguir o funcionamento de cada linha do código desenvolvido para esse exemplo.

Código comentado:

Linha(s)	Descrição
Linhas 1 e 12	Container obrigatório para todo script jQuery. Veja [C1 S1.1.6.2].
Linha 2	Define uma função a ser executada quando o usuário clica o primeiro botão denominado Coordenadas scroll.
Linha 3	Armazena, na variável `rolagemVertical`, a coordenada vertical da barra de rolagem, usando o método `scrollTop()`.
Linha 4	Idem para a coordenada horizontal.
Linha 5	Escreve dentro do elemento `span` o resultado das coordenadas da posição atual das barras de rolagem vertical e horizontal.
Linha 7	Define uma função a ser executada quando o usuário clica o segundo botão denominado Reset.
Linhas 8 e 9	Define para ambas as coordenadas da barra de rolagem o valor 0 (zero).
Linhas 10	Retira da tela o resultado das coordenadas inserido anteriomente dentro do elemento `span`.

seletorjQuery.scrollLeft()

Funciona de modo idêntico ao método `scrollTop` estudado anteriormente, retornando a posição da barra de rolagem horizontal. O `arquivo-4.2c.html` do exemplo anterior demonstra o funcionamento deste método.

seletorjQuery.scrollTop(*valor*)

Este método, quando definido com o parâmetro *`valor`*, permite movimentar a barra de rolagem vertical para a posição indicada em *`valor`*. O parâmetro é um número inteiro e a unidade de medida de *`valor`* é o pixel. O `arquivo-4.2c.html` do exemplo anterior demonstra o funcionamento deste método, quando se executa a função disparada pelo botão "Reset" que define o valor para `scrollTop` igual a 0 (zero).

seletorjQuery.scrollLeft(*valor*)

Este método, quando definido com o parâmetro *`valor`*, permite movimentar a barra de rolagem horizontal para a posição indicada em *`valor`*. O parâmetro é um número inteiro e a unidade de medida de *`valor`* é o pixel. O `arquivo-4.2c.html` do exemplo anterior demonstra o funcionamento deste método, quando se executa a função disparada pelo botão "Reset" que define o valor para `scrollLeft` igual a 0 (zero).

4.3 Largura e altura

seletorjQuery.**width()**

Acessa o valor da largura do elemento `seletorjQuery` e retorna o resultado em unidade pixel mesmo que tal largura tenha sido definida em regra CSS com outra unidade. O valor retornado não inclui as espessuras de `padding` nem `margin` porventura existentes.

No exemplo a seguir, a largura do elemento parágrafo foi definida em 20em. Observe, no arquivo que ilustra o exemplo, como o resultado é retornado em pixel. Sabe-se que por padrão 1em = 16px, logo 20em deverá retornar 320px.

Experimente aumentar o tamanho da fonte no navegador e rodar o script.

Exemplo:

```
...
<style type="text/css" media="all">
    div, p {height:50px; background:#0f0; padding:0 20px;}
    div {width:275px;}
    p {width:20em;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            var larguraDiv = $('div').width();
            $('div').after('A largura do div é de: ' + larguraDiv + ' px');
            var larguraPara = $('p').width();
            $('p').after('A largura do parágrafo é de: ' + larguraPara + ' px');
        });
    });
</script>
</head>
<body>
    <button type="button">Larguras</button><br />
    <div>DIV para determinar a largura.</div><br />
    <p>Texto de um parágrafo para determinação da largura.</p>
...
```

[arquivo-4.3a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se acessaram os valores das larguras do `div` e do `p`, os quais se armazenaram em duas variáveis denominadas `larguraDiv` e `larguraPara` respectivamente. A seguir, escreveu-se

o resultado usando o comado `after()` [C3 S3.6] para inserção de conteúdos. Veja o funcionamento desse script clicando o botão "Larguras" no arquivo indicado, disponível no site do livro.

É certo afirmar que o script desenvolvido neste exemplo, assim como em muitos dos exemplos puramente demonstrativos dos métodos jQuery, não é a melhor solução prática. Contudo, cumpre a finalidade do script, ou seja, demonstrar o funcionamento do método. Na segunda parte do livro, quando se estudarem scripts de aplicação prática, serão desenvolvidas soluções otimizadas.

seletorjQuery.width(*valor*)

Define o valor, em pixel, da propriedade CSS `width` para o elemento *`seletorjQuery`*. Isto é, define uma largura para os elementos-alvo do seletor.

Exemplo:

```
...
<style type="text/css" media="all">
   div, p {height:50px; background:#0f0; padding:0 20px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').width(350);
         $('p').width(200);
      });
   });
</script>
</head>
<body>
   <button type="button">Larguras</button><br />
   <div>DIV para definir a largura.</div><br />
   <p>Texto de um parágrafo para definir a largura.</p>
...
```

[arquivo-4.3b.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que as larguras do `div` e do `p` não foram definidas por regras CSS e assumem o valor-padrão que é igual ao valor da largura do elemento-pai, nesse caso a largura da janela. Definiram-se, com o método jQuery `width(valor)`, as larguras de 350px e 200px, respectivamente, para o `div` e o `p`. Veja o funcionamento desse script clicando o botão "Larguras" no arquivo indicado, disponível no site do livro.

seletorjQuery.height()

Acessa o valor da altura do elemento `seletorjQuery` e retorna o resultado em unidade pixel mesmo que tal altura tenha sido definida em regra CSS com outra unidade. No exemplo a seguir, a altura do elemento parágrafo foi definida em 4em. Observe, no arquivo que ilustra o exemplo, como o resultado é retornado em pixel. Sabe-se que por padrão 1em = 16px, logo 4em deverá retornar 64px.

Experimente aumentar o tamanho da fonte no navegador e rodar o script.

Exemplo:

```
...
<style type="text/css" media="all">
   div, p {width:50%; background:#0f0;}
   div {height:38px;}
   p {height:4em;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var alturaDiv = $('div').height();
         $('div').after('A altura do div é de: ' + alturaDiv + ' px');
         var alturaPara = $('p').height();
         $('p').after('A altura do parágrafo é de: ' + alturaPara + ' px');
      });
   });
</script>
</head>
<body>
   <button type="button">Alturas</button><br />
   <div>DIV para determinar a altura.</div><br />
   <p>Texto de um parágrafo para determinação da altura.</p>
...
```

[arquivo-4.3c.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se acessaram os valores das alturas do `div` e do `p`, os quais se armazenaram em duas variáveis denominadas `alturaDiv` e `alturaPara` respectivamente. A seguir, escreveu-se o resultado usando o comado `after()` [C3 S3.6] para inserção de conteúdos. Veja o funcionamento desse script clicando o botão “Alturas” no arquivo indicado, disponível no site do livro.

seletorjQuery.**height**(*valor*)

Define o valor, em pixel, da propriedade CSS `height` para o elemento *`seletorjQuery`*, isto é, define uma altura para os elementos-alvo do seletor.

Exemplo:

```
...
<style type="text/css" media="all">
   div, p {width:50%; background:#0f0;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').height(70);
         $('p').height (100);
      });
   });
</script>
</head>
<body>
   <button type="button">Alturas</button><br />
   <div>DIV para definir a altura.</div><br />
   <p>Texto de uma parágrafo para definir a altura.</p>
...
```

[arquivo-4.3d.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que as alturas do `div` e do `p` não foram definidas por regras CSS e assumem o valor-padrão `auto`, que é igual ao valor necessário para circunscrever o conteúdo inserido no elemento. Definiram-se, com o método jQuery `height(valor)`, as larguras de 70px e 100px, respectivamente, para o `div` e o `p`. Veja o funcionamento desse script clicando o botão "Alturas" no arquivo indicado, disponível no site do livro.

seletorjQuery.**outerWidth**([*booleano*])

Acessa o valor da largura do elemento *`seletorjQuery`* computando os valores dos paddings e das bordas horizontais e retorna o resultado em unidade pixel mesmo que as definições de largura e/ou `padding` tenham sido definidas em regra CSS com outra unidade. No exemplo a seguir, a largura do elemento `div` foi definida em 275px e o `padding` esquerdo e direito, 20px, totalizando um `padding` de 40px.

Observe no arquivo que ilustra o exemplo como o resultado retornado para a largura do div é de: 275px + 40px = 315px.

Para o parágrafo, o valor retornado é de: 20em x 16px + 40px = 360px.

O parâmetro booleano pode ser declarado true ou false. O valor-padrão (quando não se declara) é false. Quando declarado como true, o resultado retornado para a largura do elemento inclui também o valor das margens horizontais, se existentes.

Exemplo:

```
...
<style type="text/css" media="all">
   div, p {height:50px; background:#0f0; padding:0 20px;}
   div {width:275px;}
   p {width:20em;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var larguraDiv = $('div').outerWidth();
         $('div').after('A largura do div (incluindo o padding mais as bordas
             horizontais) é de: ' + larguraDiv + ' px');
         var larguraPara = $('p').outerWidth();
         $('p').after('A largura do parágrafo(incluindo o padding lateral) é de: '
         + larguraPara + ' px');
      });
   });
</script>
</head>
<body>
   <button type="button">Larguras</button><br />
   <div>DIV para determinação da largura + padding + borda.</div><br />
   <p>Texto de um parágrafo para determinação da largura + padding.</p>
...
```

[arquivo-4.3e.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Observe que se acessaram os valores das larguras do div e do p, os quais se armazenaram em duas variáveis denominadas larguraDiv e larguraPara respectivamente. A seguir, escreveu-se o resultado usando o comado after() [C3 S3.6] para inserção de conteúdos. Veja o funcionamento desse script clicando o botão “Larguras” no arquivo indicado, disponível no site do livro.

seletorjQuery.outerHeight([*booleano*])

Acessa o valor da altura do elemento e funciona de maneira idêntica ao método `outerWidth()` explicado anteriormente.

seletorjQuery.innerWidth()

Acessa o valor da largura do elemento incluindo o valor de padding horizontal.

seletorjQuery.innerHeight()

Acessa o valor da altura do elemento incluindo o valor do padding horizontal.

Figura ilustrativa

Observe, na figura 4.1, um diagrama do Box Model CSS esclarecendo os valores de offset e as larguras retornadas pelos métodos estudados anteriormente.

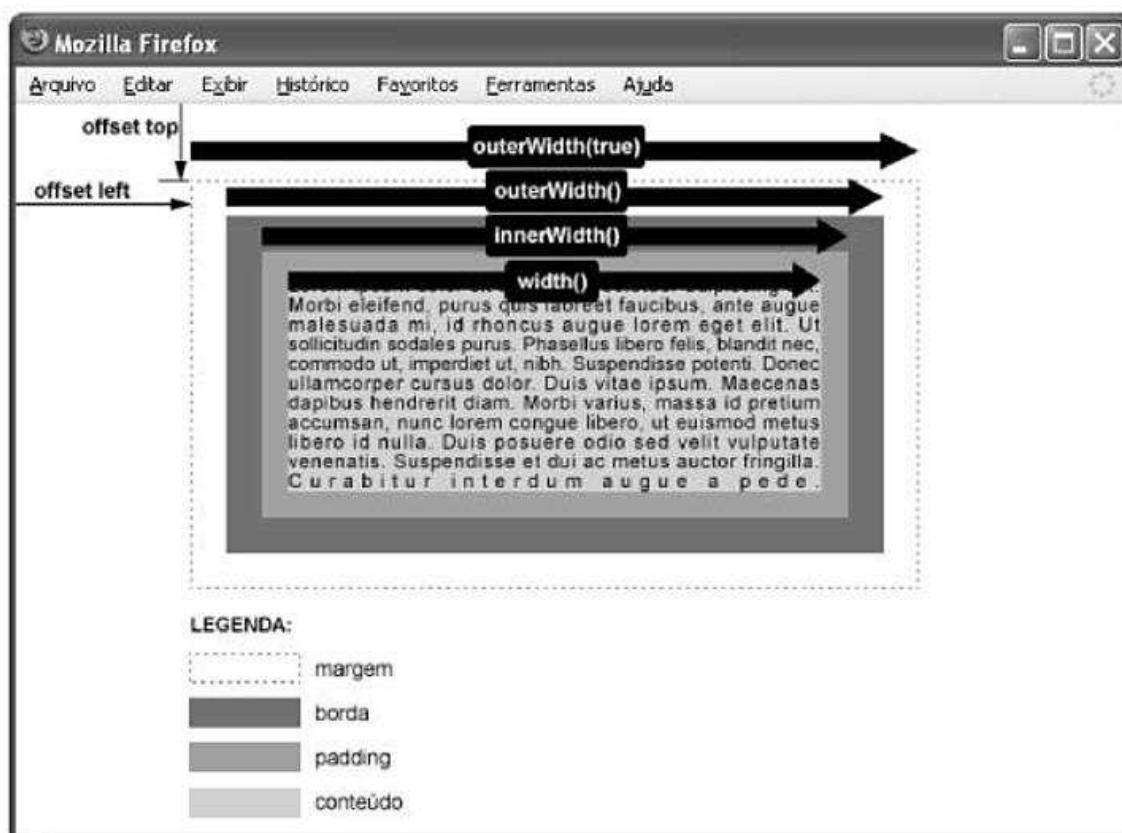



Figura 4.1 – Métodos dimensionais.

Os valores no sentido vertical, não mostrados na figura 4.1, são retornados de maneira semelhante pelos métodos para altura.

Os métodos dimensionais foram incluidos na biblioteca a partir da versão jquery-1.2.6. Em versões anteriores, para pleno suporte a estes métodos, é necessário a instalação de um plug-in denominado `dimensions.js`.

4.4 Inspeção do DOM

seletorjQuery.**filter(***filtro***)**

Acessa todos os elementos `seletorjQuery` e seleciona somente aqueles definidos em `filtro`.

No exemplo a seguir, acessaram-se todos os parágrafos e selecionaram-se somente aqueles cujo atributo `class` é `alvo`, aplicando neles a cor vermelha.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('p').filter('.alvo').css('color', 'red');
        });
    });
</script>
</head>
<body>
    <button type="button">Filtrar</button><br />
    <p>Texto de parágrafo sem atributo classe.</p>
    <p class="alvo">Texto de parágrafo com atributo class = "alvo".</p>
    <p class="alvo">Texto de parágrafo com atributo class = "alvo".</p>
    <p>Texto de parágrafo sem atributo classe</p>
    <p class="alvo">Texto de parágrafo com atributo class = "alvo".</p>
...
```

[arquivo-4.4a.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão “Filtrar” no arquivo indicado, disponível no site do livro.

seletorjQuery.**filter**(*função*)

Este método funciona de forma idêntica ao método anterior, admitindo para filtro uma função JavaScript.

seletorjQuery.**is**(*expressão*)

Acessa todos os elementos `seletorjQuery` e verifica se pelo menos um deles satisfaz as condições estabelecidas no parâmetro *expressão*. Retorna os booleanos `true`/`false` caso positivo ou negativo respectivamente.

No exemplo a seguir, verifica-se se algum `div` possui a classe de nome `cor` e escreve-se o resultado em um parágrafo com o método `text()`. Como existe tal `div`, o valor retornado deve ser `true`.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         var resultados = ($('div').is('.cor'));
         $('p').text('O valor retornado é: ' + resultados);
      });
   });
</script>
</head>
<body>
   <button type="button">Resultado</button><br />
   <div>Primeiro div</div>
   <div class="cor">Segundo div</div>
   <div>Terceiro div</div>
   <p></p>
...
```

[arquivo-4.4b.html]

Nesse exemplo, utiliza-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão “Resultado” no arquivo indicado, disponível no site do livro.

seletorjQuery.**not**(*expressão*)

Acessa todos os elementos `seletorjQuery` e exclui da seleção aqueles que satisfazem a condição estabelecida em `expressão`.

No exemplo a seguir, selecionaram-se os divs e aplicou-se uma cor de fundo naqueles que não possuem a classe de nome `sem-fundo`.

Exemplo:

```
...
<style type="text/css" media="all">
   div {float:left; margin:10px; width:90px; height:90px; border:1px solid #000;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').not('.sem-fundo').css('backgroundColor', '#ff0');
      });
   });
</script>
</head>
<body>
   <button type="button">Resultado</button><br />
   <div class="sem-fundo">Primeiro div</div>
   <div>Segundo div</div>
   <div>Terceiro div</div>
   <div class="sem-fundo">Quarto div</div>
   <div>Quinto div</div>
...
```

[arquivo-4.4c.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão "Resultado" no arquivo indicado, disponível no site do livro.

seletorjQuery.**slice**(*início* [*fim*])

Acessa todos os elementos `seletorjQuery` e seleciona desde o elemento na posição `inicio` até o elemento na posição `fim`. Não inclui o elemento da posição `fim`. O parâmetro `fim` é opcional e, se omitido, faz a seleção até o último elemento.

Lembre-se de que a contagem JavaScript inicia em 0 (zero), assim `início` = 3 e `fim` = 11 seleciona do quarto ao décimo primeiro elemento.

No exemplo a seguir, selecionaram-se os divs da posição dois à posição cinco e aplicou-se uma cor de fundo neles.

Exemplo:

```
...
<style type="text/css" media="all">
   div {float:left; margin:10px; width:90px; height:90px; border:1px solid #000;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').slice(1, 5).css('backgroundColor', '#ff0');
      });
   });
</script>
</head>
<body>
   <button type="button">Resultado</button><br />
   <div class="sem-fundo">Primeiro div</div>
   <div>Segundo div</div>
   <div>Terceiro div</div>
   <div class="sem-fundo">Quarto div</div>
   <div>Quinto div</div>
   <div>Sexto div</div>
...
```

[arquivo-4.4d.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão "Resultado" no arquivo indicado, disponível no site do livro.

seletorjQuery.**add**(*expressão*)

Seleciona todos os elementos `seletorjQuery` e a estes adiciona os elementos definidos no parâmetro `expressão`.

No exemplo a seguir, selecionaram-se os divs e a estes se adicionou o parágrafo com a classe de nome `clear` aplicando uma cor de fundo no conjunto selecionado (os divs mais o parágrafo com a classe `clear`).

Exemplo:

```
...
<style type="text/css" media="all">
   div {float:left; margin:10px; width:90px; height:90px; border:1px solid #000;}
   .clear {clear:both;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('div').add('.clear').css('backgroundColor', '#ff0')
      });
   });
</script>
</head>
<body
   <button type="button">Resultado</button><br />
   <div>Primeiro div</div>
   <div>Segundo div</div>
   <div>Terceiro div</div>
   <p class="clear">Primeiro parágrafo</p>
   <p>Segundo parágrafo</p>
   <p>Terceiro parágrafo</p>
...
```

[arquivo-4.4e.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão "Resultado" no arquivo indicado, disponível no site do livro.

seletorjQuery.**children**([*expressão*])

Seleciona todos os elementos-filho de `seletorjQuery`. O parâmetro `expressão` é facultativo e, quando definido, destina-se a filtrar os elementos-filho encontrados.

No exemplo a seguir, selecionaram-se todos os elementos-filho do parágrafo, filtraram-se os `span` e neles se aplicou uma cor de fundo.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
```

```
            $('p').children('span').css('backgroundColor', '#ff0')
        });
    });
</script>
</head>
<body>
    <button type="button">Resultado</button><br />
    <p>Parágrafo com <span>span um</span><b>bold um</b>
    e a seguir <i>itálico um</i>.<br /> Na continuação
    <span>span dois</span> e finalmente <span>último span.</span></p>
...
```

[arquivo-4.4f.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão "Resultado" no arquivo indicado, disponível no site do livro.

seletorjQuery.**find**([*expressão*])

Seleciona todos os elementos descendentes de `seletorjQuery` que satisfazem `expressão`.

No exemplo a seguir, encontraram-se os elementos em itálico pertencentes ao parágrafo e aplicou-se uma cor de fundo ao conjunto selecionado.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {
            $('p').find('i').css('backgroundColor', '#ff0')
        });
    });
</script>
</head>
<body>
    <button type="button">Resultado</button><br />
    <p>Parágrafo com <span>span um</span><b>bold um</b>
    e a seguir <i>itálico um</i>.<br /> Na continuação
    <span>span dois</span> e finalmente <span>último span.</span></p>
...
```

[arquivo-4.4g.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão "Resultado" no arquivo indicado, disponível no site do livro.

Note que este método cumpre a mesma função do seletor composto `$(p span)`. A vantagem de seu uso está no fato de aceitar uma expressão como parâmetro de busca, o que não é possível com seletores compostos.

seletorjQuery.**parent([*expressão*])**

Seleciona todos os elementos que sejam elemento-pai do *seletorjQuery*. Admite o parâmetro *expressão* como um filtro.

No exemplo a seguir, selecionaram-se os elementos-pai de parágrafo e aplicou-se uma cor de fundo ao conjunto selecionado.

Exemplo:

```
...
<style type="text/css" media="all">
   div, blockquote {width:300px; height:40px; margin:10px; border:1px solid #000;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('p').parent().css('backgroundColor', '#ff0')
      });
   });
</script>
</head>
<body>
   <button type="button">Resultado</button><br />
   <div><h4>Cabeçalho h4 filho de um div</h4></div>
   <div><p>Parágrafo filho de um div</p></div>
   <blockquote><p>Parágrafo filho de blockquote</p></blockquote>
...
```

[arquivo-4.4h.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão "Resultado" no arquivo indicado, disponível no site do livro.

***seletorjQuery*.parents([*expressão*])**

Seleciona todos os ancestrais do *seletorjQuery*. Admite o parâmetro *expressão* como um filtro.

No exemplo a seguir, selecionaram-se todos os elementos ancestrais do elemento span e aplicou-se uma borda preta de 2px de espessura em volta de cada um deles. Para facilitar a visualização deste exemplo no arquivo disponível no site do livro, acresentaram-se regras CSS para definir uma cor de fundo para cada um dos elementos ancestrais de span.

Exemplo:

```
...
<style type="text/css" media="all">
   html {background:#0f0;}
   body {background:#ccc; width:400px;}
   div {margin:20px; background:#ffc;}
   blockquote {background:#0f0;}
   p {background:#0ff;}
   span {background:#f93;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').click(function() {
         $('span').parents().css('border', '2px solid #000')
      });
   });
</script>
</head>
<body>
   <div>
      <blockquote>
         <p>Parágrafo
            <span>
               filho
            </span>
         de blockquote.
         </p>
      </blockquote>
   </div>
<button type="button">Resultado</button>
...
```

[arquivo-4.4i.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando o botão “Resultado” no arquivo indicado, disponível no site do livro.

seletorjQuery.**prev**([*expressão*])

Seleciona o elemento adjacente imediatamente anterior ao `seletorjQuery`. Admite o parâmetro *expressão* como filtro.

No exemplo a seguir, o script parte do último elemento e a cada clique no botão disparador do evento seleciona o elemento-irmão anterior definindo para este uma imagem de fundo. Ao atingir o primeiro elemento, o script para de selecionar porque o primeiro elemento não possui elemento anterior.

Exemplo:

```
...
<style type="text/css" media="all">
   div {float:left; width:80px; height:80px; margin:10px; border:2px solid #00f;}
   button {display:block;clear:both;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      var divCorrente = $('.corrente');
      divCorrente.css('background', 'url(flor.jpg) no-repeat');
      $('button').click(function() {
         divCorrente = divCorrente.prev();
         $('div').css('background', '');
         $divCorrente.css('background', 'url(flor.jpg) no-repeat');
      });
   });
</script>
</head>
<body>
   <div>DIV um</div>
   <div>DIV dois</div>
   <div>DIV três</div>
   <div>DIV quatro</div>
   <div>DIV cinco</div>
   <div>DIV seis</div>
   <div class="corrente">DIV sete</div>
   <button type="button">&laquo; Anterior</button>
...
```

[arquivo-4.4j.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando seguidamente o botão "Anterior" no arquivo indicado, disponível no site do livro.

seletorjQuery.**next**([*expressão*])

Seleciona o elemento adjacente imediatamente posterior ao `seletorjQuery`. Admite o parâmetro *expressão* como filtro.

Este método funciona da mesma maneira que o anterior, selecionando elementos em ordem crescente.

Esse script está no arquivo indicado a seguir.

 [arquivo-4.4k.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando seguidamente o botão "Próximo" no arquivo indicado, disponível no site do livro.

seletorjQuery.**prevAll**([*expressão*])

Seleciona todos os elementos-irmão adjacentes `seletorjQuery` anteriores a ele. Admite o parâmetro *expressão* como filtro.

Este método funciona da mesma maneira que `prev()`, selecionando todos os elementos-irmão adjacentes anteriores.

Este script está no arquivo indicado a seguir.

 [arquivo-4.4l.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando seguidamente o botão "Anterior" no arquivo indicado, disponível no site do livro.

seletorjQuery.**nextAll**([*expressão*])

Seleciona todos os elementos-irmão adjacentes de `seletorjQuery`. Admite o parâmetro *expressão* como filtro.

Este método funciona da mesma maneira que `prevAll()`, selecionando todos os elementos-irmão adjacentes subsequentes.

Este script está no arquivo indicado a seguir.

 [arquivo-4.4m.html]

Nesse exemplo, utilizou-se um botão para disparar o evento. Veja o funcionamento desse script clicando seguidamente o botão "Próximo" no arquivo indicado, disponível no site do livro.

seletorjQuery1.....seletorjQueryN.**end()**.*método_continua()*

Este método permite interferir em um encadeamento de métodos jQuery, revertendo a seleção ao objeto anterior ao último objeto adjacente ao método `end()` na cadeia. Na sintaxe mostrada, os efeitos contidos no método *`método_continua()`* serão aplicados ao *`seletorjQuery(N-1)`*.

Os objetos jQuery aos quais se aplica este método são `add`, `andSelf`, `children`, `filter`, `find`, `map`, `next`, `nextAll`, `not`, `parent`, `parents`, `prev`, `prevAll`, `siblings` e `slice` e também à função `clone`.

Exemplo:

```
...
<style type="text/css" media="all">
   div {width:400px; height:100px; margin:10px; border:2px solid #00f;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button:eq(0)').click(function() {
         $('div').find('p').css('background', '#f00');
      });    // sem uso de end() a cor do fundo é para o parágrafo dentro do div

      $('button:eq(1)').click(function() {
         $('div').find('p').end().css('background', '#f00');
      }); // mesmo código com uso de end() a cor do fundo reverte para o div
      $('button:eq(2)').click(function() {
         $('div, p').css('background', '');
      });
   });
</script>
</head>
<body>
   <button type="button">Sem end()</button>
   <button type="button">Com end()</button>
   <button type="button">Reset</button>
   <div>
      <p>Este parágrafo está dentro do div</p>
   </div>
...
```

 [arquivo-4.4n.html]

Nesse exemplo, mostraram-se duas situações com o objetivo de enfatizar na prática o funcionamento de `end()`. No primeiro caso, a estilização fez-se diretamente no parágrafo selecionado pelo script e, na segunda situação, usando o método `end()`, transferiu-se a estilização para o elemento `div` imediatamente antes do método que seleciona o parágrafo no encadeamento.

seletorjQuery1.....seletorjQueryN.**andSelf()**.*método_continua()*

Este método, tal como o anterior, permite interferir em um encadeamento de métodos jQuery, revertendo a seleção aos dois últimos objetos da cadeia. Na sintaxe mostrada, os efeitos contidos no método *`método_continua()`* serão aplicados simultaneamente ao *`seletorjQuery(N-1)`* e ao *`seletorjQueryN`*.

Os objetos jQuery aos quais se aplica este método são `add`, `andSelf`, `children`, `filter`, `find`, `map`, `next`, `nextAll`, `not`, `parent`, `parents`, `prev`, `prevAll`, `siblings` e `slice` e também à função `clone`.

Exemplo:

```
...
<style type="text/css" media="all">
   div {width:400px; height:100px; margin:10px; border:2px solid #00f;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button:eq(0)').click(function() {
         $('div').find('p').andSelf().css('border', '2px dotted #f00');
      });

      $('button:eq(1)').click(function() {
         $('div, p').css('border', '');
      });
   });
</script>
</head>
<body>
   <button type="button">andSelf()</button>
   <button type="button">Reset</button>
   <div>
      <p>Este parágrafo está dentro do div</p>
   </div>
...
```

seletorjQuery.**siblings([*expressão*])**

Seleciona todos os irmãos de `seletorjQuery`. Admite o parâmetro `expressão` como filtro.

No exemplo a seguir, o script seleciona todos os irmãos do elemento `li` cuja classe é `diferente` e aplica uma cor de fundo amarela. Note que o elemento `li` com a classe `diferente` não é selecionado, mas somente seus irmãos.

Exemplo:

```
...
<script type="text/javascript">
    $(document).ready(function() {
        $('button:eq(0)').click(function() {
            $('li.diferente').siblings().css('background', 'yellow');
        });

        $('button:eq(1)').click(function() {
            $('li.diferente').siblings().css('background', '');
        });
    });
</script>
</head>
<body>
    <button type="button">sibilings()</button>
    <button type="button">Reset</button>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li class="diferente">Item 3</li>
        <li>Item 4</li>
        <li>Item 5</li>
    </ul>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
        <li>Item 5</li>
    </ul>
...
```

[arquivo-4.4p.html]

seletorjQuery.**map(***função***)**

Transforma o conjunto de objetos retornado pelo `seletorjQuery` em um array cujos valores podem ser elementos ou referências. O `parâmetro` função define como será feita a seleção dos valores do array e qual a sua natureza.

No exemplo a seguir, a `função` recolhe o valor de cada um dos campos de um formulário, armazena em um array e escreve os valores.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button:eq(0)').click(function() {
         $('div').append($('input').map(function() {
            return $(this).val();
         }).get().join(', '));
      });

      $('button:eq(1)').click(function() {
         $('div').empty();
      });
   });
</script>
</head>
<body>
   <p>Digite nome e e-mail nos campos e rode o script</p>
   <button type="button">map()</button>
   <button type="button">Reset</button>
   <form action="" method="get">
      <label>Nome:<br /><input type="text" /></label><br />
      <label>E-mail:<br /><input type="text" /></label><br />
      <span>Sexo:</span><br />
      <label><input type="radio" name="sexo" value="masculino" /> Masculino</label><br />
      <label><input type="radio" name="sexo" value="feminino" /> Feminino</label>
   </form>
      <div></div>
...
```

[arquivo-4.4q.html]

CAPÍTULO 5

Eventos

Neste capítulo, serão estudados os eventos disponíveis na biblioteca jQuery. Na linguagem JavaScript, a entrada em funcionamento de um script depende de um evento. É lícito afirmar que se não houvesse eventos, simplesmente não haveria JavaScript. Evento, também chamado de gatilho, é uma ação do usuário que desencadeia o início do script ou faz rodá-lo.

Quando o usuário abre uma página web, clica um botão, pressiona uma tecla, seleciona um campo etc., ocorre o evento. JavaScript possui mecanismos capazes de detectar a ação do usuário e desencadear o script. O conhecimento da sintaxe que captura um evento é fundamental para o desenvolvimento de scripts.

5.1 Eventos auxiliares

seletorjQuery.blur(*função*)

Quando o `seletorjQuery` perde o foco, ocorre o evento `blur()` e entra em ação o script definido no parâmetro `função`. Esse evento é amplamente usado em campos de formulário para disparar um script (por exemplo: um script para validar os dados inseridos no campo) quando o usuário abandona o campo.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('input').blur(function() {
         $('input').css('background', 'yellow');
```

```
            $('.mensagem').text('Você disparou o evento blur.');
        });
    });
</script>
</head>
<body>
    <p>Clique dentro do campo de texto abaixo<br />
    e depois em qualquer lugar fora dele.</p>
    <input type="text" />
    <p class="mensagem"></p>
...
```

[arquivo-5.1a.html]

> Os eventos jQuery seguem uma sintaxe ligeiramente diferente dos da linguagem JavaScript. Note que o evento jQuery `blur` tem seu correspondente JavaScript com sintaxe `onblur`. Se você desenvolve JavaScript e algo der errado com seu script que usa eventos, antes de mais nada verifique a grafia dos eventos.

seletorjQuery.change(*função*)

O evento `change()` ocorre quando o valor em um campo de formulário perde o foco em favor de outro valor. O uso clássico desse evento é em campos de formulário do tipo `select`, no qual o evento ocorre quando o usuário seleciona uma das opções.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('select').change(function() {
            var valorEscolhido = $('option:selected').text();
            $('.mensagem').html('Você disparou o evento change.<br />Selecionou: '
            + valorEscolhido);
        });
    });
</script>
</head>
<body>
    <p>Faça algumas seleções no campo select.</p>
    <select>
        <option>Escolha um</option>
        <option>Escolha dois</option>
        <option>Escolha três</option>
        <option>Escolha quatro</option>
```

```
        <option>Escolha cinco</option>
        <option>Escolha seis</option>
        <option>Escolha sete</option>
        </select>
    <p class="mensagem"></p>
...
```

[arquivo-5.1b.html]

seletorjQuery.click(*função*)

O evento `click()` ocorre quando o usuário clica o ponteiro de um dispositivo apontador (por exemplo: o ponteiro do mouse) sobre *seletorjQuery*, disparando o script definido no parâmetro *função*. Nos exemplos constantes dos capítulos anteriores, nos scripts em que se usou um botão para fazê-los funcionar, utilizou-se o evento click. Volte a um daqueles exemplos para estudar esse evento.

seletorjQuery.dblclick(*função*)

O evento `dblcllick()` ocorre quando o usuário clica duas vezes seguidas o ponteiro de um dispositivo apontador (por exemplo: o ponteiro do mouse) sobre *seletorjQuery*, disparando o script definido no parâmetro *função*.

Exemplo:

```
...
<style type="text/css" media="all">
    div {
        width:90px;
        height:90px;
        background:green;
        border:2px solid #000;
        }
    .muda-cor {background:red;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div').dblclick(function() {
            $('div').toggleClass('muda-cor');
            $('.mensagem').html('Você disparou o evento dblclick.');
        });
    });
</script>
</head>
```

```
<body>
   <p>Um duplo clique no div e ocorre o evento dblclick.</p>
   <div></div>
   <p class="mensagem"></p>
...
```

[arquivo-5.1c.html]

seletorjQuery.error(*função*)

O evento `error()` ocorre quando um erro é detectado, quer seja de programação (por exemplo: sintaxe errada), quer seja com um elemento na árvore do documento. Um exemplo de erro com o elemento `img` ocorre quando o caminho para a imagem definido no atributo `src` está errado, é inválido ou quando não existe a imagem. Nesses casos, o navegador Internet Explorer renderiza um pequeno ícone com aproximadamente 30 x 30px indicando quebra da imagem. Para esconder o ícone, use o script mostrado a seguir.

```
$('img').error(function() {
   $(this).hide();
});
```

Não há uma diretriz pública para tratamento de erros em JavaScript, o que torna o assunto complexo, na medida em que envolve implementações proprietárias. O estudo e aprofundamento do tratamento de erros fogem ao escopo deste livro.

seletorjQuery.focus(*função*)

Quando o *`seletorjQuery`* recebe o foco, ocorre o evento `focus()` e entra em ação o script definido no parâmetro *função*. Esse evento é amplamente usado em campos de formulário para disparar um script (por exemplo: um script mostrando um pop-up com dicas para o preenchimento do campo) quando o usuário entra no campo.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('input').focus(function() {
         $('input').css('background', 'yellow');
         $('.mensagem').text('Use apenas números para informar seu CPF.');
      });
   });
</script>
```

```
</head>
<body>
    <p>Clique dentro do campo de texto abaixo.</p>
    <label>CPF: <input type="text" /></label>
    <p class="mensagem"></p>
...
```

[arquivo-5.1d.html]

seletorjQuery.keydown(*função*)

Ocorre o evento `keydown()` quando o usuário pressiona uma tecla qualquer de seu teclado. Esse evento tem sua aplicação mais comum em campos de formulário, particularmente em `textarea`, para contar e informar ao usuário o número de caracteres digitados quando se pretende limitar a extensão da entrada (por exemplo: em campos para entrada de comentários ou envio de mensagens).

Se você pretende bloquear a entrada de caracteres a partir de um determinado teto, use jQuery somente para informar ao usuário o número de caracteres à medida que ele digita. Para efetuar o bloqueio a partir do teto, use uma programação que rode no servidor (por exemplo: PHP, ASP, ColdFusion etc.).

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('input').keydown(function() {
            $('body').css('background', '#cff');
            $('.mensagem').html('Ao pressionar uma tecla qualquer do seu teclado<br />
            ocorreu o evento keydown e a cor de fundo da página mudou.');
        });
    });
</script>
</head>
<body>
    <p>Entre uma letra qualquer no campo abaixo.</p>
    <input type="text" />
    <p class="mensagem"></p>
...
```

 [arquivo-5.1e.html]

seletorjQuery.**keyup**(*função*)

Ocorre o evento `keyup()` quando o usuário solta uma tecla qualquer do teclado que tenha sido pressionada.

seletorjQuery.**keypress**(*função*)

Ocorre o evento `keypress()` quando o usuário realiza a sequência apertar e soltar uma tecla qualquer do teclado.

seletorjQuery.**load**(*função*)

Ocorre o evento `load()` quando o *`seletorjQuery`* termina de ser carregado e nesse momento entra em ação o script definido no parâmetro *`função`*. Esse evento, normalmente, é usado para o objeto `window` e ocorre quando toda a página acaba de ser carregada.

A sintaxe para esse evento é mostrada a seguir:

```
$('window').load (function() {
    // seu script aqui
});
```

seletorjQuery.**mousedown**(*função*)

O evento `mousedown()` ocorre quando o usuário pressiona o botão de um dispositivo apontador (por exemplo: o ponteiro do mouse) sobre *`seletorjQuery`*, disparando o script definido no parâmetro *`função`*.

Exemplo:

```
...
<style type="text/css" media="all">
    body {cursor:pointer;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('h2').mousedown(function() {
            $(this).append(' mousedown ');
        });
    });
</script>
</head>
```

```
<body>
   <h2>Faça acontecer mousedown aqui.</h2>
...
```

[arquivo-5.1f.html]

seletorjQuery.mouseup(*função*)

O evento `mouseup()` ocorre quando o usuário solta o botão de um dispositivo apontador (por exemplo: o ponteiro do mouse) sobre *seletorjQuery*, disparando o script definido no parâmetro *função*.

Exemplo:

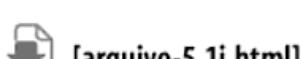
```
...
<style type="text/css" media="all">
   body {cursor:pointer;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('h2').mouseup(function() {
         $(this).append(' mouseup ');
      });
   });
</script>
</head>
<body>
   <h2>Faça acontecer mouseup aqui.</h2>
...
```

[arquivo-5.1g.html]

O script a seguir demonstra o emprego conjunto dos dois eventos anteriores, facilitando a visualização do momento em que cada um deles ocorre.

```
...
<style type="text/css" media="all">
   h2 {cursor:pointer;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('h2').mousedown(function() {
         $(this).append(' <span style="color:blue">mousedown</span> ');
      }).mouseup(function() {
         $(this).append(' <span style="color:red">mouseup</span> ');
```

```
        });
    });
</script>
</head>
<body>
    <h2>Clique, segure e solte.</h2>
...
```

[arquivo-5.1h.html]

seletorjQuery.mouseover(*função*)

O evento `mouseover()` ocorre quando o usuário passa um dispositivo apontador (por exemplo: o ponteiro do mouse) sobre *seletorjQuery*, disparando o script definido no parâmetro *função*.

Exemplo:

```
...
div {width:90px; height:90px; background:green; border:2px solid black;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div').mouseover(function() {
            $(this).css('background', 'red');
        });
    });
</script>
</head>
<body>
    <p>Passe o mouse sobre o div</p>
    <div>DIV</div>
...
```

[arquivo-5.1i.html]

seletorjQuery.mouseout(*função*)

O evento `mouseout()` ocorre quando o usuário retira o dispositivo apontador (por exemplo: o ponteiro do mouse) de cima do *seletorjQuery*, disparando o script definido no parâmetro *função*.

Exemplo:

```
...
div {width:90px; height:90px; background:green;  border:2px solid black;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div').mouseout(function() {
            $(this).css('background', 'red');
        });
    });
</script>
</head>
<body>
    <p>Coloque o mouse sobre o div e retire</p>
    <div>DIV</div>
...
```

[arquivo-5.1j.html]

O script a seguir demonstra o emprego conjunto dos dois eventos anteriores. Observe que ao combinar os métodos `mouseover()` e `mouseout()`, obtém-se o clássico efeito rollover, tão usado em menus de navegação.

```
...
div {width:90px; height:90px; background:green; border:2px solid black;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div').mouseover(function() {
            $(this).css('background', 'red');
        }).mouseout(function() {
            $(this).css('background', 'green');;
        });
    });
</script>
</head>
<body>
    <p>Passe o mouse sobre o div e retire.</p>
    <div>DIV</div>
...
```

 [arquivo-5.1k.html]

seletorjQuery.mousemove(*função*)

O evento `mousemove()` ocorre quando o usuário movimenta o dispositivo apontador (por exemplo: o ponteiro do mouse) sobre o *seletorjQuery*, disparando o script definido no parâmetro *função*. Esse evento é amplamente usado para armazenar as coordenadas do dispositivo apontador.

Exemplo:

```
...
<style type="text/css" media="all">
   div {margin-left:100px;width:200px; height:200px;background:red;
      border:2px solid black;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('div').mousemove(function(o) {
         var windowCoordenadas = '(x = ' + o.clientX + ' y = ' + o.clientY + ')';
         $('span').css('display', 'block').text('Coordenadas: ' + windowCoordenadas);
         $('div').mouseout(function() {
            $('span').css('display', 'none');
         });
      });
   });
</script>
</head>
<body>
   <p>Movimente o mouse sobre o div.</p>
      <div></div>
      <span></span>
...
```

[arquivo-5.1l.html]

seletorjQuery.resize(*função*)

O evento `resize()` ocorre quando há redimensionamento do *seletorjQuery*, disparando o script definido no parâmetro *função*. No exemplo a seguir, o script é executado quando o usuário redimensiona a janela do navegador.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
```

```
    $(document).ready(function() {
        $(window).resize(function() {
            $('body'). text('Ops! Você redimensionou a janela do navegador.');
        });
    });
</script>
</head>
<body>
    <span></span>
</body>
</html>
...
```

 [arquivo-5.1m.html]

Para visualizar o funcionamento do script, abra o arquivo correspondente em seu navegador e redimensione a janela do navegador.

seletorjQuery.scroll(*função*)

O evento `scroll()` ocorre quando o usuário aciona uma das barras de rolagem do *`seletorjQuery`*, disparando o script definido no parâmetro *`função`*.

Exemplo:

```
...
<style type="text/css" media="all">
    div {width:200px; height:150px; overflow:scroll;}
    b {color:red;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div').scroll(function() {
            $('b').text('Você acionou a barra de rolagem do div');
        });
    });
</script>
</head>
<body>
    <div>
    <h4>Div com barra de rolagem</h4>
    <p>...muito conteúdo...</p>
    </div>
    <b></b>
...
```

 [arquivo-5.1n.html]

seletorjQuery.**select**(*função*)

O evento `select()` ocorre quando o usuário seleciona um texto ou fragmento de texto do *seletorjQuery*, disparando o script definido no parâmetro *função*. Mas, atenção: esse evento só é válido para seleções de textos contidos em campos de formulário destinados à entrada de textos, tais como `input` e `textarea`. Não confunda esse evento com o evento `change()`, que ocorre quando o usuário muda a seleção em um campo de formulário do tipo `select`.

Exemplo:

```
...
<style type="text/css" media="all">
   span {font:small-caps bold 20px sans-serif; color:#f00;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $(document).select(function() {
         $('<span>Você disparou o evento select</span>')
         .show().fadeOut(1500).appendTo('div');
      });
   });
</script>
</head>
<body>
   <p>Clique e arraste o mouse no texto dentro do campo para selecioná-lo.</p>
   <input type="text" value="selecione este texto">
   <div></div>
...
```

[arquivo-5.1o.html]

seletorjQuery.**submit**(*função*)

O evento `submit()` ocorre quando o usuário submete ou envia os dados colhidos ou contidos em *seletorjQuery*, disparando o script definido no parâmetro *função*. Esse evento é amplamente empregado para verificação dos dados preenchidos pelo usuário em um formulário tão logo ele tente enviá-lo.

Exemplo:

```
...
<style type="text/css" media="all">
   div {color:#f00;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
```

```
<script type="text/javascript">
   $(document).ready(function() {
      $('form').submit(function() {
         if ($('input:first').val() == '') {
            $('div').text('Por favor! Preencha o campo.');
            return false;
         }
         if ($('input:first').val() !== 'jQuery') {
            $('div').text('Lamento! entrou a palavra errada.');
            return false;
         } else {
            $('div').text('Parabéns! entrou a palavra certa.');
         }
      });
   });
</script>
</head>
<body>
   <p>Entre a palavra jQuery no campo abaixo e clique o botão Enviar.<br />
   Atenção: campo sensível ao tamanho de caixa</p>
   <form action="javascript:void()">
   <label>Campo de texto</label><br />
   <input type="text" />
   <input type="submit" value="Enviar" />
   </form>
   <div></div>
...
```

[arquivo-5.1p.html]

seletorjQuery.unload(*função*)

Ocorre o evento `unoad()` quando o usuário abandona o *`seletorjQuery`* que tenha sido carregado anteriormente e nesse momento entra em ação o script definido no parâmetro *`função`*. Esse evento é usado para o objeto `window` e ocorre quando o usuário fecha uma página.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $(window).unload(function() {
         alert('Esta é uma demonstração do evento unload.');
      });
   });
```

```
</script>
</head>
<body>
   <p>Feche esta janela para constatar o evento unload em ação.</p>
</body>
...
```

[arquivo-5.1q.html]

5.2 Eventos de interação

seletorjQuery.**toggle**(*f1 f2* [*f3...fN*])

O evento `toggle()` permite executar duas, e opcionalmente mais, funções, seguidas e alternadamente mediante cliques sucessivos no botão de um dispositivo apontador (por exemplo: o ponteiro do mouse) sobre o *`seletorjQuery`*, fazendo entrar em ação os scripts definidos nas funções *`f1`* e *`f2`*. Quando o usuário clica uma vez no *`seletorjQuery`*, é executada a função *`f1`*, clicando outra vez, é executada a função *`f2`*, mais uma vez, é executada a função *`f1`* e assim sucessiva e indefinidamente alternando a execução das funções.

Esse evento admite a opção de adicionar mais de dois scripts e nesse caso a execução deles se faz de *`f1`* até *`fN`*, volta a *`f1`* e repete-se o ciclo indefinidamente.

Exemplo:

```
...
<style type="text/css" media="all">
   p {cursor:pointer;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('p').toggle(
      function() {
         $('img').css('display', 'block').attr({src:'2.gif'});
      },
      function() {
         $('img').css('display', 'block').attr({src:'3.gif'});
      },
      function() {
         $('img').css('display', 'block').attr({src:'4.gif'});
      },
      function() {
```

```
            $('img').css('display', 'block').attr({src:'1.gif'});
        })
    });
</script>
</head>
<body>
    <p>Clique seguidamente aqui,<br />
    para ver o evento toggle() em ação.</p>
    <img alt="Bate coração" src="1.gif" />
...
```

[arquivo-5.2a.html]

seletorjQuery.**hover**(*sobre fora*)

O evento `hover()` permite executar duas funções: uma quando o usuário coloca um dispositivo apontador (por exemplo: o ponteiro do mouse) sobre o *seletorjQuery* e a outra quando ele retira o dispositivo apontador de sobre o *seletorjQuery*, fazendo entrar em ação os scripts definidos nas funções *sobre* e *fora*. Este método, tal como o método `toggle()`, pode ser usado para obter o conhecido efeito rollover.

Exemplo:

```
...
<style type="text/css" media="all">
    img {cursor:pointer;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('img').hover(
        function() {
            $('img').css('display', 'block').attr({src:'2-over.gif'});
        },
        function() {
            $('img').css('display', 'block').attr({src:'2.gif'});
        })
    });
</script>
</head>
<body>
    <p>Passe o mouse sobre o coração,<br />
    para ver o evento hover() em ação.</p>
    <img alt="Bate coração" src="2.gif" />
...
```

[arquivo-5.2b.html]

5.3 Manipuladores de eventos

seletorjQuery.**bind**(*tipo* [*dados*] *função*)

O manipulador de eventos `bind()` permite vincular um evento ao *`seletorjQuery`*, fazendo entrar em ação os scripts definidos nos parâmetros *`função`*. O parâmetro *`dados`* é facultativo e o parâmetro *`tipo`* define o tipo de evento a vincular. Os eventos possíveis são: `blur`, `focus`, `load`, `resize`, `scroll`, `unload`, `click`, `dblclick`, `mousedown`, `mouseup`, `mousemove`, `mouseover`, `mouseout`, `mouseenter`, `mouseleave`, `change`, `select`, `submit`, `keydown`, `keypress`, `keyup` e `error`.

Exemplo:

```
...
<style type="text/css" media="all">
    div {cursor:pointer; width:100px; height:100px; border:2px solid #000;
    span {display:block;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div').bind('mouseover', function() {
            $(this).css('borderWidth', '10px');
            $('span:eq(0)').text('Ocorreu o evento mouseover. Aumentou espessura da borda.');
        });

        $('div').bind('click', function(o) {
            var coordenadas = 'x= ' + o.pageX + ', y= ' + o.pageY;
            $('span:eq(1)').text('Ocorreu o evento click nas coordenadas: ' + coordenadas);
        });

        $('div').bind('dblclick', function() {
            $(this).css('background', 'red');
            $('span:eq(2)').text('Ocorreu o evento dblclick. Mudou a cor de fundo.');
        });
    });
</script>
</head>
<body>
    <p>Eventos vinculados com bind():</p>
        <ul>
            <li>mouseover - aumenta espessura das bordas.</li>
            <li>click - amazena e mostra as coordenadas do click.</li>
            <li>dblclick - troca cor de fundo.</li>
        </ul>
```

```
    <div></div>
        <span></span>
        <span></span>
        <span></span>
...
```

[arquivo-5.3a.html]

seletorjQuery.**one**(*tipo* [*dados*] *função*)

O manipulador de eventos `one()` permite vincular um ou mais eventos ao *`seletorjQuery`*, fazendo entrar em ação os scripts definidos nos parâmetros *função*. O parâmetro dados é facultativo e o parâmetro tipo define o tipo de evento a vincular. Aqui cada um dos elementos selecionados por *`seletorjQuery`* executa o script uma única vez. No exemplo a seguir, a caixa de alerta será mostrada somente ao primeiro clique em cada parágrafo. O segundo clique não executa mais o script.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('p').one('click', function() {
            alert($(this).text());
        });
    });
</script>
</head>
<body>
    <p>Primeiro parágrafo</p>
    <p>Segundo parágrafo</p>
    <p>Terceiro parágrafo</p>
    <p>Quarto parágrafo</p>
...
```

[arquivo-5.3b.html]

Para visualizar a diferença de funcionamento, faça uma cópia do arquivo que contém esse script e substitua `one()` por `bind()`. Carregue e página modificada e clique várias vezes cada parágrafo.

seletorjQuery.**trigger(***tipo* [*dados*]**)**

O manipulador de eventos `trigger()` permite que um evento vinculado a um seletor *seletorjQuery* seja vinculado simultaneamente a um segundo *seletorjQuery*, sendo possível fazer entrar em ação ambos os scripts com o evento ocorrendo somente no seletor *seletorjQuery* vinculado. No exemplo a seguir, foram projetados dois botões. Um clique no botão um dispara o script um e um clique no botão dois dispara o script dois e também o um, pois se utilizou o manipulador de eventos `trigger()` para vincular o script um aos botões um e dois.

Exemplo:

```
...
<style type="text/css" media="all">
   span {color:red;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button:first').click(function() {
         atualizaClique($('span:first'));
      });

      $('button:last').click(function() {
         $('button:first').trigger('click');
         atualizaClique($('span:last'));
      });

      function atualizaClique(o) {
         var n = parseInt(o.text(), 0);
         o.text(n + 1);
      }
   });
</script>
</head>
<body>
   <button type="button">Botão n.1</button>
   <button type="button">Botão n.2</button>
   <p>Soma dos cliques nos botões 1 e 2 = <span>0</span></p>
   <p>Cliques somente no botão n.2 = <span>0</span></p>
...
```

[arquivo-5.3c.html]

Para iniciantes: a função `atualizaCampo(o)` funciona como um contador, pois recebe um parâmetro, no exemplo dado, o valor contido em um elemento span, armazena-o em uma variável e acrescenta uma unidade. Os dois scripts rodam a função `atualizaCampo(o)` a cada clique, reescrevendo no elemento span o novo valor da variável n.

seletorjQuery.unbind(*tipo* [*dados*])

O manipulador de eventos `unbind()` funciona de forma inversa a `bind()` estudado anteriormente, assim permite desvincular um ou mais eventos que tenham sidos vinculados ao *`seletorjQuery`*.

5.4 Notas sobre eventos

O sistema de eventos na biblioteca jQuery foi construído em conformidade com as Recomendações do W3C. Isto significa a normatização do objeto evento e a garantia da devida passagem do evento para seu respectivo manipulador, sem necessidade de fazer verificações do tipo `window.event`. Também se normatizaram os alvos dos eventos e as propriedades para coordenadas das páginas `page.X` e `page.Y`.

Outra funcionalidade ligada a eventos é a disponibilização, na biblioteca, dos métodos `stopPropagation()` e `preventDefault()`.

event.type

Este método retorna uma string que descreve a natureza do evento.

Exemplo:

```
$('a').click(function(event) {
   alert(event.type);
)};
```

Retorna:

```
"click"
```

event.target

Este método retorna o alvo do evento, ou seja, o elemento ao qual se atribuiu a função de disparar o evento ou um de seus elementos-filho.

Exemplo:

```
$('a[href=http://www.maujor.com]').click(function(event) {
   alert(event.target);
   return false;
   });
```

Retorna:

```
"http://www.maujor.com/"
```

event.pageX/Y

Este método retorna as coordenadas do ponteiro do dispositivo apontador (por exemplo: o mouse) em relação ao documento.

Exemplo:

```
$('div').click(function(event) {
   alert('Você clicou na posição: x = ' + event.pageX + " px, y = " + event.pageY
   +' px');
});
```

Retorna:

```
"As coordenadas do click"
```

Para visualizar os três eventos descritos anteriormente, veja

 [arquivo-5.4a.html]

event.preventDefault()

Este método impede o navegador de seguir o comportamento-padrão para a ação. Por exemplo: ao clicar um link, o navegador é impedido de seguir para o endereço do link, ou, ainda, ao clicar um botão de submissão de um formulário, este não é submetido. O equivalente JavaScript para esse método é: `return false`.

Exemplo:

```
$('a[href=http://www.maujor.com]').click(function(event) {
   event.preventDefault();
});
```

event.stopPropagation()

Este método impede que o evento anexado a um elemento se propague para os ancestrais do elemento. É conhecido na programação JavaScript como "efeito bolha". Por tratar-se de um efeito que depende de situações particulares, apenas se informará que existe a funcionalidade de se poder cancelar o efeito bolha existente na biblioteca, ficando a critério do desenvolvedor usá-la quando estiver diante de problemas causados por propagação indevida de eventos. A sintaxe geral é mostrada a seguir:

```
$('p').click(function(event) {
   event.stopPropagation();
   //script vai aqui;
});
```

CAPÍTULO 6

Efeitos

Neste capítulo, serão tratados os métodos para obtenção de efeitos. São os efeitos um dos objetivos do emprego de jQuery. As tarefas de esconder e revelar conteúdos e fazer funcionar um menu de uma forma visualmente agradável com transições suaves e surpreendentes são uma funcionalidade da linguagem que certamente irá enriquecer a experiência do usuário. E, com um planejamento bem feito, você conseguirá implementar efeitos sem bloquear o acesso ou prejudicar a usabilidade.

A apresentação dos métodos para obtenção de efeitos mostrados neste capítulo segue uma metodologia diferente da adotada nos capítulos anteriores. Assim ocorreu porque os scripts que demonstram os efeitos são curtos e resolveu-se reunir os efeitos de mesma natureza em um arquivo único. Essa metodologia, como vantagem adicional, irá possibilitar-lhe fazer a comparação entre os efeitos consultando um único arquivo.

6.1 Efeitos básicos

seletorjQuery.show()

O efeito `show()` revela abruptamente o elemento *`seletorjQuery`* que tenha sido escondido anteriormente, quer com uso de script, quer com regra CSS.

Exemplo de sintaxe:

```
$('div').show()
```

Para tirar o máximo proveito de seus estudos, após conhecer todos os efeitos básicos descritos neste capítulo, consulte os arquivos desenvolvidos para demonstrar cada um deles, disponíveis para download no site do livro.

seletorjQuery.show(*velocidade* [*função*])

O método `show(velocidade [função])` causa o mesmo efeito criado com o método anterior, porém o uso do parâmetro `velocidade` permite revelar suavemente o elemento `seletorjQuery`.

O parâmetro `velocidade` admite três valores definidos por palavra-chave, que são: `slow`, `normal` e `fast`, para obtenção de velocidades de revelações lenta, normal e rápida respectivamente. De modo opcional, a velocidade pode ser definida por um número representando o tempo de revelação em milissegundos.

O parâmetro `função` é opcional e permite definir uma função cuja execução será desencadeada tão logo o efeito termine.

Exemplos de sintaxes:

```
$('div').show('slow')
```

ou

```
$('div').show(1000)
```

ou

```
$('div').show('fast', function() {
   // seu script aqui;
});
```

seletorjQuery.hide()

O método `hide()` causa o efeito contrário ao criado pelo método `show()`, escondendo abruptamente o elemento `seletorjQuery` que tenha sido revelado.

Exemplo de sintaxe:

```
$('div').hide()
```

seletorjQuery.hide(*velocidade* [*função*])

O método `hide(velocidade [função])` causa o mesmo efeito criado pelo método anterior, porém o uso do parâmetro *velocidade* permite esconder suavemente o elemento *seletorjQuery*.

O parâmetro *velocidade* admite três valores definidos por palavras-chave, que são: `slow`, `normal` e `fast`, para obtenção de velocidades de revelações lenta, normal e rápida respectivamente. De modo opcional, a velocidade pode ser definida por um número representando o tempo de revelação em milissegundos.

O parâmetro *função* é opcional e permite definir uma função cuja execução será desencadeada tão logo o efeito termine.

Exemplos de sintaxes:

```
$('div').hide('slow')
```

ou

```
$('div').hide(1000)
```

ou

```
$('div').hide('fast', function() {
   // seu script aqui;
   });
```

seletorjQuery.toggle()

O método `toggle()` permite criar o efeito de alternância de visibilidade do conjunto de elementos definido por *seletorjQuery*. Os elementos invisíveis tornam-se visíveis e vice-versa a cada disparo do evento.

Exemplo de sintaxe:

```
$('button').click(function() {
      $('p').toggle();
   });
```

Consulte o arquivo referenciado a seguir, para ver todos os efeitos básicos estudados anteriomente em ação:

```
<style type="text/css" media="all">
   div {width:200px; height:300px;margin:30px 0;border:2px solid #000;color:#fff;
      background:#f00; display:none;}
</style>
```

```
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').eq(0).click(function() {
            $('div').show();
        });
        $('button').eq(1).click(function() {
            $('div').hide();
        });
        $('button').eq(2).click(function() {
            $('div').show('slow');
        });
        $('button').eq(3).click(function() {
            $('div').hide('slow');
        });
        $('button').eq(4).click(function() {
            $('div').show(3000);
        });
        $('button').eq(5).click(function() {
            $('div').hide(1500);
        });
        $('button').eq(6).click(function() {  // efeito toggle()
            $('p').toggle();
        });
    });
</script>
</head>
<body>
    <button type="button">Efeito show()</button>
    <button type="button">Efeito hide()</button><br /><br />
    <button type="button">Efeito show('slow')</button>
    <button type="button">Efeito hide('slow')</button><br /><br />
    <button type="button">Efeito show(3000)</button>
    <button type="button">Efeito hide(1500)</button><br /><br />
    <button type="button">Efeito toggle()</button><br /><br />>
        <div>DIV em todo seu esplendor.</div>
        <p style="display:none; width:300px; background:#0f0;">
            Clique seguidamente o botão Efeito toggle.
        </p>
...
```

[arquivo-6.1a.html]

6.2 Efeitos corrediços

seletorjQuery.**slideDown**(*velocidade* [*função*])

O método para criar o efeito `slideDown()` destina-se a revelar suavemente o elemento *`seletorjQuery`* que tenha se escondido anteriormente, fazendo a transição de invisível para visível por meio do aumento gradativo da altura do *`seletorjQuery`* escondido. A altura do elemento vai sendo revelada de cima para baixo.

O parâmetro *`velocidade`* admite três valores definidos por palavras-chave, que são: `slow`, `normal` e `fast`, para obtenção das velocidades de revelações lenta, normal e rápida respectivamente. De modo opcional, a velocidade pode ser definida por um número representando o tempo de revelação em milissegundos.

O parâmetro *`função`* é opcional e permite definir uma função cujo script entrará em ação tão logo *`seletorjQuery`* seja revelado.

Exemplo de sintaxe:

```
$('div').slideDown('slow')
```

Primeiro conheça todos os efeitos corrediços a seguir e, depois, consulte a página indicada, disponível para download no site do livro, para testá-los na prática.

seletorjQuery.**slideUp**(*velocidade* [*função*])

O método `slideUp()` destina-se a criar o efeito de esconder o elemento *`seletorjQuery`* que tenha se revelado anteriormente. A altura do elemento é escondida no sentido de baixo para cima.

Essa animação apresenta o que parece ser um bug, pelo menos até a versão atual da biblioteca. Quando se aplica o efeito em um elemento para o qual se tenha definido um `padding` vertical, o efeito esconder ignora o `padding` como sendo integrante da altura total do elemento. Isso causa um salto ao final da animação, pois ao ser recolhida uma amplitude igual à altura do elemento menos o valor do `padding`, este desaparece bruscamente. Na página constante do site do livro, que demonstra esse efeito, mostra-se o suposto bug.

Para solucionar o bug, em vez de declarar um `padding` para o elemento, crie um `div` interno ao elemento e defina para ele margens verticais. Definir margens para um container interno tem o mesmo efeito que definir `padding` para o container externo.

Exemplo de sintaxe:

```
$('div').slideUp('slow')
```

seletorjQuery.**slideToggle**(*velocidade* [*função*])

O método `slideToggle()` destina-se a causar um efeito de alternância de visibilidade do conjunto de elementos definido por *`seletorjQuery`* com o uso de efeitos causados por `slideDown()` e `slideUP()`. Os elementos invisíveis tornam-se visíveis e vice-versa, alternadamente, a cada disparo do evento.

Exemplo de sintaxe:

```
$('div').slideToggle(1500)
```

Consulte o arquivo que contém o script mostrado a seguir para ver todos os efeitos corrediços estudados anteriomente em ação:

```
<style type="text/css" media="all">
   div {width:200px; height:300px; margin:30px 0; border:2px solid #000;
      color:#fff; background:#f00; display:none;}
   .bug {height:140px; padding:80px 0;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button').eq(0).click(function() {
         $('div').eq(0).slideDown('slow');
      });
      $('button').eq(1).click(function() {
         $('div').eq(0).slideUp('slow');
      });
      $('button').eq(2).click(function() {
         $('.bug').slideDown('slow');
      });
      $('button').eq(3).click(function() {
         $('.bug').slideUp('slow');
      });
      $('button').eq(4).click(function() {
         $('.alterna').slideToggle(1500);
      });
   });
```

```
</script>
</head>
<body>
    <button type="button">Efeito slideDown('slow')</button>
    <button type="button">Efeito slideUp('slow')</button><br /><br />
    <button type="button">BUG slideDown('slow')</button>
    <button type="button">BUG slideUp('slow')</button><br /><br />
    <button type="button">Efeito slideToggle(1500)</button>
        <div>DIV em todo seu esplendor.</div>
        <div class="bug">DIV em todo seu esplendor.<br />
        Com declaração de padding</div>
        <div class="alterna">Demonstração do efeito slideToggle(1500).<br />
        Clique seguidamente no botão Efeito slideToggle().</div>
...
```

[arquivo-6.2a.html]

6.3 Efeitos de opacidade

seletorjQuery.**fadeIn(***velocidade* [*função*]**)**

O método `fadeIn()` destina-se a criar o efeito de revelar o elemento *`seletorjQuery`* que se tenha escondido anteriormente, fazendo a transição de invisível para visível por meio do aumento gradativo da opacidade do *`seletorjQuery`* escondido. O elemento vai sendo revelado por mudança de opacidade de 0 (invisível) a 100% (opaco ou visível).

O parâmetro *`velocidade`* admite três valores definidos por palavras-chave, que são: `slow`, `normal` e `fast`, para obtenção de velocidades de revelações lenta, normal e rápida respectivamente. De modo opcional, a velocidade pode ser definida por um número representando o tempo de revelação em milissegundos.

O parâmetro *`função`* é opcional e permite definir uma função cujo script entrará em ação tão logo *`seletorjQuery`* seja revelado.

Exemplo de sintaxe:

```
$('div').fadeIn('slow')
```

seletorjQuery.**fadeOut(***velocidade* [*função*]**)**

O método `fadeOut()` destina-se a criar o efeito de esconder o elemento *`seletorjQuery`* que esteja visível, fazendo a transição de visível para invisível por meio da diminuição gradativa da opacidade do *`seletorjQuery`* visível. O elemento vai sendo escondido por mudança da opacidade de 100% (opaco ou visível) a 0 (invisível).

seletorjQuery.**fadeTo(***velocidade opacidade* [*função*]**)**

O método `fadeTo()` destina-se a alterar as condições de visibilidade do elemento *`seletorjQuery`*, fazendo uma mudança gradativa da opacidade atual para a opacidade definida no parâmetro *opacidade*. Esse parâmetro deve ser definido por um número compreendido entre 0 e 1, sendo 0 invisível e 1 opaco.

Consulte o arquivo referenciado a seguir, para ver todos os efeitos de opacidade estudados anteriomente em ação.

```
<style type="text/css" media="all">
    div {width:150px; height:150px; position:absolute;left:30px; top:80px;
        margin:30px 0; border:2px solid #000; background:#0f0; display:none;}
    .fadeto {display:block; background:red; left:200px;}
    p {position:absolute; width:130px; left:210px; top:100px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('button').eq(0).click(function() {
            $('div').eq(0).fadeIn(1500);
        });
        $('button').eq(1).click(function() {
            $('div').eq(0).fadeOut('slow');
        });
        $('button').eq(2).click(function() {
            $('div').eq(1).fadeTo(1500, 0.3);
        });
        $('button').eq(3).click(function() {
            $('div').eq(1).fadeTo(1500, 1);
        });
    });
</script>
</head>
<body>
    <p>Ops tem algo aqui! Tudo que está atrás deste div
    é revelado quando se diminui sua opacidade. :-) </p>
    <button type="button">Efeito fadeIn(1500)</button>
    <button type="button">Efeito fadeOut('slow')</button><br /><br />
    <button type="button">Efeito fadeTo(1500, 0.3)</button>
    <button type="button">Efeito fadeTo(1500, 1)</button>
    <div>Demonstração de fadeIn()/fadeOut.</div>
    <div class="fadeto"></div>
...
```

[arquivo-6.3a.html]

6.4 Efeitos personalizados

seletorjQuery.**animate**(*definições* [*velocidade*] [*aceleração*] [*função*])

seletorjQuery.**animate**(*definições* [*opções*])

O método `animate()` permite-lhe criar animações personalizadas para o *`seletorjQuery`*. Conforme mostrado, existem duas sintaxes para esse método.

O princípio básico que rege o funcionamento desse método é o fato de que somente propriedades CSS de valor numérico são possíveis de serem animadas. Por exemplo: você pode criar uma animação que faça um `div` crescer ou encolher em largura ou altura, alterando gradativamente suas propriedades CSS, `width` ou `height`. Contudo, com o uso desse método, não poderá criar uma animação que altere gradativamente a cor de fundo do `div`, pois a propriedade CSS `background-color` não é numérica. Os valores, para as propriedades CSS a animar, configuradas no parâmetro *`definições`*, são os valores finais da animação, isto é, a animação se faz desde o valor corrente da propriedade até o valor definido. Por exemplo: se o valor inicial da largura é `width:200px` e você define em sua animação personalizada o parâmetro `width:50px`, trata-se de uma animação por encolhimento.

A partir de jQuery versão 1.2, permite-se usar unidades CSS `em` e `%`(onde aplicável). Além disso, os operadores de incremento e decremento `+=` e `-=` são válidos para criar animações incrementais, como `{left:'+=10px'}`, que define uma coordenada horizontal esquerda gradativamente crescente em saltos de 10px.

Existem vários plug-ins que permitem criar animações, além das permitidas por padrão pela biblioteca, como descrito anteriormente. Não é o escopo deste livro examinar tais plug-ins. Serão destacadas as animações-padrão que, em resumo, são aquelas que manipulam as propriedades `top`, `left`, `width`, `height` e `opacity`, além de outras possíveis de estilização numérica, como `font-size` e `border-width`. Além de valores numéricos, permite-se também especificar o valor das propriedades com o uso de uma das seguintes palavras-chave: `hide` `show` e `toggle`. Por exemplo:

```
$('div').click(function() {
    $(this).animate(
        {opacity:'toggle',
        width:'hide'},
        {duration:'2000'}
    );
});
```

O parâmetro `velocidade` é opcional e admite três valores definidos por palavras-chave, que são: `slow`, `normal` e `fast`, para obtenção de velocidades de animações lenta, normal e rápida respectivamente. De modo opcional, a velocidade pode ser definida por um número representando o tempo da animação em milissegundos.

O parâmetro `aceleração` é opcional e requer o uso de plug-in para funcionar. Esse parâmetro define como será a aceleração da animação. Os valores `linear` e `swing` são nativos.

O parâmetro `função` é opcional e permite definir uma função cujo script entrará em ação tão logo a animação termine.

Exemplo:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('button:eq(0)').click(function() {
      $('div').animate(
         {width:'300px',
         height:'600px',
         opacity:0.33,
         border:'10px dashed #f00'   },
         1500)
      });
   });
</script>
</head>
<body>
   <button type="button">Animar</button>
   <div></div>
...
```

[arquivo-6.4a.html]

A sintaxe alternativa permite-lhe definir um parâmetro denominado *opções* que são objetos para a animação. Essa variante da sintaxe admite também definir uma *velocidade*, uma *aceleração*, uma chamada para a *função* a ser executada tão logo a animação se complete e um parâmetro denominado *queue* que controla o início da animação. Nos exemplos práticos que serão estudados na segunda parte do livro, o uso dessa sintaxe será mais bem abordado.

seletorjQuery.**stop()**

O método `stop()` permite que você interrompa uma animação em andamento.

Exemplo:

```
...
<style type="text/css" media="all">
   div.interna {position:absolute; width:80px; height:80px; border:2px solid #000;
               background:#f00; left:10px; top:100px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('button:eq(0)').click(function() {
         $('div').animate({left:'+=1000px'}, 8000);
      });
      $('button:eq(1)').click(function() {
         $('div').stop();
      });
      $('button:eq(2)').click(function() {
         $('div').animate({left:'-=1000px'}, 8000);
      });
   });
</script>
</head>
<body>
   <button type="button">Animar</button>
   <button type="button">Parar</button>
   <button type="button">Voltar</button>
   <div class="interna"></div>
...
```

[arquivo-6.4b.html]

CAPÍTULO 7

Funções utilitárias

Neste capítulo, serão examinadas as funções utilitárias disponíveis na biblioteca jQuery. Uma função é dita utilitária quando desenvolvida para atender a necessidades específicas de um projeto. Em geral, uma função utilitária, ao contrário de um método jQuery, destina-se a manipular objetos JavaScript que não sejam elementos do DOM ou realizar operações não específicas a objetos.

7.1 Introdução

Nos capítulos anteriores, estudaram-se os métodos jQuery. Denominou-se genericamente de métodos jQuery todos os métodos e funções possíveis de serem aplicados a um conjunto de elementos do DOM. Por exemplo: quando se estuda o método para adicionar estilos aos parágrafos de um documento, mostra-se a sintaxe a seguir:

```
$('p').css({propriedade:'valor', propriedade:'valor', ...})
```

O construtor `$()` seleciona um conjunto de elementos, no caso do exemplo os parágrafos constantes da árvore do documento – o DOM –, e neles aplica as regras CSS definidas no método `css()`.

Convém ressaltar a característica de encadeamento própria da biblioteca jQuery que permite encadear métodos, unindo-os com um ponto (.). Na cadeia formada pela união de métodos com um ponto, cada adição de um método retorna um objeto pronto para receber novo método, em um processo sem fim.

Outro aspecto importante a não esquecer é a característica de seleção múltipla automática, isto é, a seleção feita pelo construtor `$()` compreende um conjunto

de elementos-alvo, sem necessidade de criar loops de seleção, como ocorre na linguagem JavaScript formal.

Para fins de estudo, as funções utilitárias serão classificadas em dois grupos:

- funções utilitárias nativas da biblioteca jQuery;
- funções utilitárias personalizadas.

As funções utilitárias para uso personalizado dependem das particularidades de cada desenvolvimento e, por isso mesmo, fogem do escopo deste livro. Serão enfatizadas as funções utilitárias nativas da jQuery. Contudo, ao final do capítulo, uma simples função utilitária personalizada será desenvolvida com o objetivo de mostrar a técnica de criação de tais funções.

7.2 Flags para agentes de usuário

Tecnicamente, as flags para agentes de usuário são variáveis e não propriamente funções. Destinam-se a detectar informações sobre o navegador usado pelo usuário.

Os scripts de detecção das características do navegador foram amplamente empregados em construção de sites, com a finalidade de tomar decisões de como servir códigos em função do suporte oferecido por navegadores de diferentes fabricantes.

Esta é uma prática em desuso segundo os modernos conceitos de desenvolvimento. Não se irá discorrer longamente sobre as razões para evitar o uso de scripts de detecção, mas vale lembrar um dos princípios fundamentais dos Padrões Web que estabelece a criação de códigos indepedentes de navegadores e plataformas.

Então, é lícito perguntar: por que jQuery inclui em sua biblioteca essas funcionalidades de detecção? A resposta é simples: a ferramenta está disponível, evite usá-la, esgote todas as alternativas a seu uso, mas se seu projeto reveste-se de condicionantes e características que justifiquem seu uso, utilize-as.

jQuery.browser

Compreende um conjunto de flags referentes à versão e à família do navegador. As flags são as relacionadas a seguir.

Flag	Descrição
msie	Retorna true se o navegador é identificado como Microsoft Internet Explorer.
mozilla	Retorna true se o navegador é identificado como Firefox, Camino, Netscape ou outro baseado no Mozilla.
safari	Retorna true se o navegador é identificado como Safari, OmniWeb ou outro baseado no Safari.
opera	Retorna true se o navegador é identificado como Opera.
version	Retorna a versão da engine de renderização do navegador.

Convém ressaltar que a detecção se faz para uma determinada família de navegadores e não para um navegador específico. A detecção se faz por famílias baseadas no pressuposto de que navegadores pertencentes a uma mesma família possuem características idênticas. A maioria dos navegadores modernos enquadra-se em uma das quatro famílias citadas anteriormente.

A flag `version` detecta a versão da engine de renderização e não a versão do navegador, como pode parecer à primeira vista.

No exemplo a seguir, apresentamos um script para a detecção das flags `version` e família do navegador:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('.versao').click(function() {
         var $versaoNavegador = $.browser.version;
         $('p').append('A versão da engine de renderização deste navegador é: <b>'
            + $versaoNavegador + '</b><br />');

         if (jQuery.browser.mozilla) {
            $('p').append('Este navegador pertence à família: <b>Mozilla</b><br />')}
         else if (jQuery.browser.msie) {
            $('p').append('Este navegador pertence à família:
               <b>Microsoft Internet Explorer</b><br />')}
         else if (jQuery.browser.safari) {
            $('p').append('Este navegador pertence à família: <b>Safari</b><br />')}
         else if (jQuery.browser.opera) {
            $('p').append('Este navegador pertence à família: <b>Opera</b><br />')
         } else {
            $('p').append('Não foi possível identificar a que família pertence
                este navegador.')};
         });
```

```
        $('.reset').click(function () {
            $('p').empty();
        });
    });
</script>
</head>
<body>
    <button type="button" class="versao">Flags deste navegador</button>
    <button type="button" class="reset">Reset</button><br />
    <p></p>
...
```

[arquivo-7.2a.html]

jQuery.boxModel

Denomina-se Box Model um modelo-padrão de renderização ou apresentação visual de um box segundo a formatação CSS preconizada pelas recomendações do W3C. Observe, na figura 7.1, um diagrama esclarecendo o Box Model.

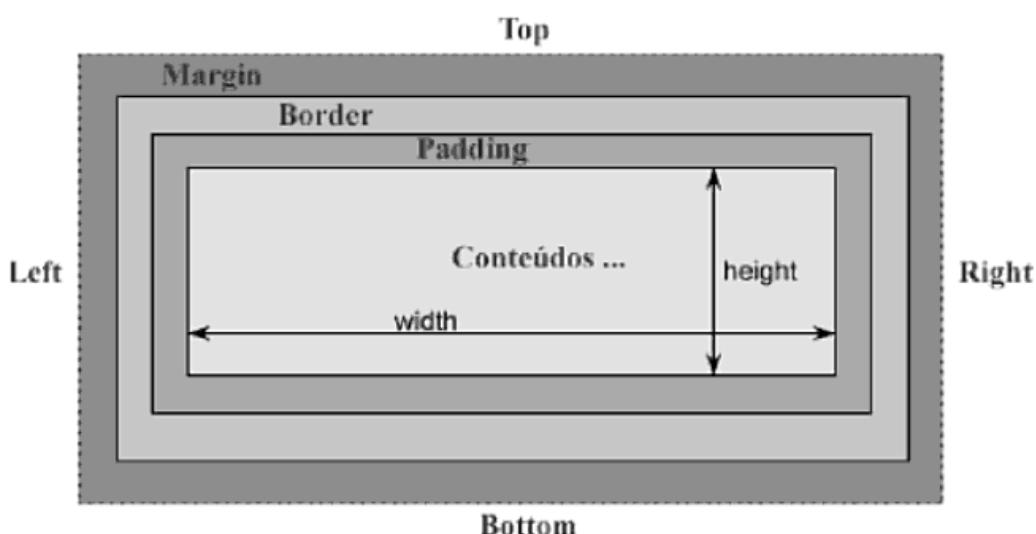



Figura 7.1 – Box Model.

Neste diagrama, destacou-se uma área mais interior denominada área dos conteúdos, cujas dimensões (largura e altura) são definidas pelas propriedades CSS `width` e `height`. Segue-se uma área denominada enchimento, cuja espessura é definida pela propriedade CSS `padding`.

Em volta do enchimento, há uma borda cujas espessura e cor e cujo tipo são definidos pela propriedade CSS `border`. Finalmente, há um espaço denominado margem, com espessura definida pela propriedade CSS `margin`.

A área da margem é sempre transparente. As dimensões da área de conteúdos dependem de uma série de fatores, entre eles, definição explícita de dimensões, natureza e tipo de conteúdo. A propriedade CSS `background` define o fundo a ser aplicado nas áreas de conteúdos, de enchimento e da borda.

Este é o modelo-padrão do W3C. Contudo, os navegadores Internet Explorer, versões para Windows anteriores à versão 6 e também as versões 6 e 7, quando renderizam em modo kirks, não seguem o modelo-padrão e adotam uma formatação conhecida como "Box Model quebrado".

Diz-se que um navegador renderiza em modo kirks quando da marcação HTML não consta a declaração de DOCTYPE ou há uma declaração de DOCTYPE em desacordo com aquelas padronizadas pelo W3C. Em oposição ao modo kirks, há a chamada renderização em modo standard.

O Box Model quebrado considera as dimensões CSS `width` e `height` como totais, ou seja, nelas estão incluídos os valores de `padding` e `border`. As margens continuam exteriores ao box.

Observe, na figura 7.2, um diagrama dos dois Box Models.

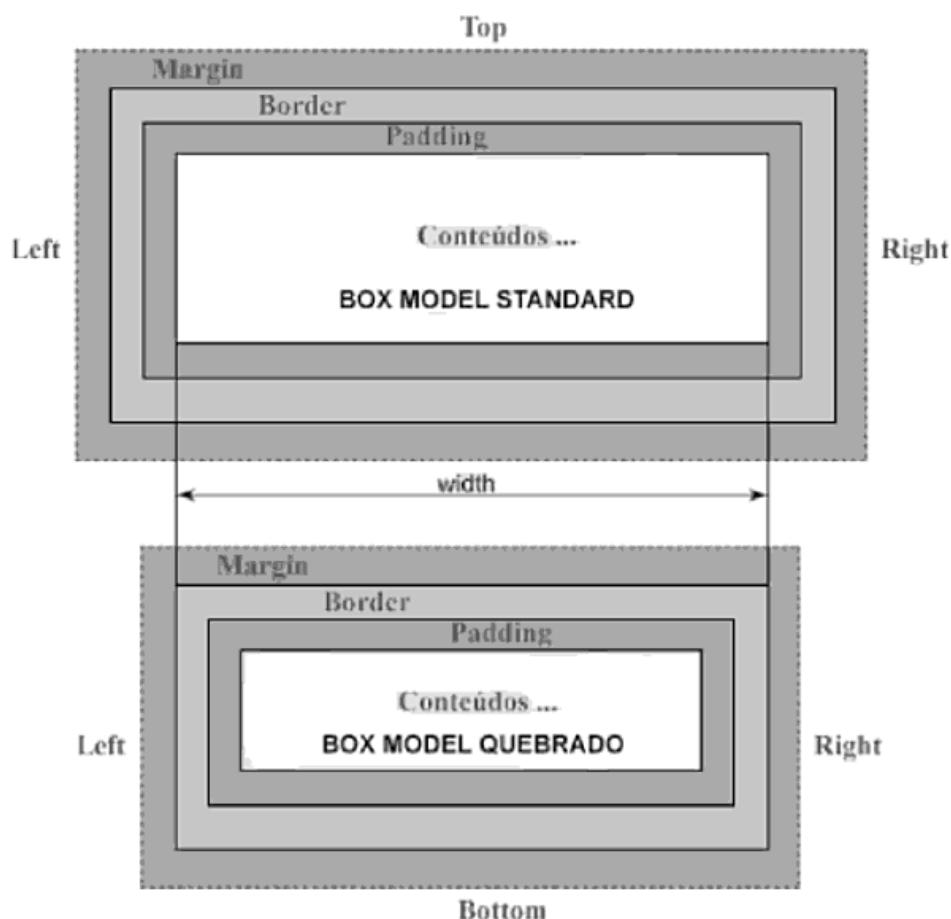



Figura 7.2 – Box Models standard e quebrado.

No exemplo a seguir, há um script para a detecção do Box Model adotado pelo navegador:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('.versao').click(function() {
      if (jQuery.boxModel == true) {
         $('p').append('Este navegador adota o Box Model: <b>Standard</b>.')}
      else if (jQuery.boxModel == false) {
         $('p').append('Este navegador adota o Box Model: <b>Quebrado</b>.')
      } else {
         $('p').append('Não foi possível identificar o Box Model adotado por este
              navegador.')};
      });

      $('.reset').click(function() {
         $('p').empty();
      });
   });
</script>
</head>
<body>
   <button type="button" class="versao">Box Model deste navegador</button>
   <button type="button" class="reset">Reset</button><br />
   <p></p>
...
```

[arquivo-7.2b.html]

Faça uma cópia do `arquivo-7.2b.html` mostrado no exemplo e retire a declaração de DOCTYPE da marcação. Visualize a cópia em qualquer versão do Internet Explorer, rode o script de detecção do Box Model e observe o valor retornado para o Box Model.

7.3 Operações com arrays e objetos

jQuery.each(*objeto função*(*chave* ou *índice valor*))

Esta função utilitária permite fazer interações tanto por um conjunto de elementos da árvore do documento como pelos elementos de um array. O parâmetro *objeto* define uma coleção de objetos ou um array no qual será feita a interação e o parâmetro *função* define a ação sobre cada um dos itens da interação.

Esta função difere do método `each()` estudado em [C2 S2.1], pois enquanto tal método faz interações exclusivamente em objetos jQuery, esta fução utilitária tem uma abrangência mais ampla, permitindo interação em qualquer entidade da linguagem, incluindo arrays.

O parâmetro *`função`* admite dois parâmetros: o primeiro parâmetro é uma *`chave`* (key), no caso de interações por objetos, ou um *`índice`* (index), no caso de interações por arrays, e o segundo parâmetro, o *`valor`* da chave ou índice.

Para exemplificar, será reescrito o script para detecção da versão e família do navegador desenvolvido no `arquivo-7.2a.html` mostrado anteriormente. Nesse script, utiliza-se uma série de métodos condicionais para testar a família do navegador. Usar a função `each()` fornece uma solução bem mais elegante e concisa.

Assim, aplica-se interação no objeto `jQuery.browser` que retornará os pares `versão:número` e `família:booleano true/false`.

Exemplo:

```
...
<script type="text/javascript">
    $(document).ready(function() {
        $('.versao').click(function() {
            jQuery.each(jQuery.browser, function(i, val) {
                $('p').append(i + ' : <b>' + val + '</b><br />');
            });
        });
        $('.reset').click(function() {
            $('p').empty();
        });
    });
</script>
</head>
<body>
    <button type="button" class="versao">Flags deste navegador</button>
    <button type="button" class="reset">Reset</button><br />
    <p></p>
...
```

[arquivo-7.3a.html]

jQuery.grep(*array função*(*valor índice*)[*inverter*])

Esta função permite fazer interações pelos elementos de um array e filtrá-los. O parâmetro *array* define o array no qual será feita a interação e o parâmetro *função* define uma ação de filtragem dos elementos do array.

O parâmetro *função* admite dois parâmetros: o primeiro parâmetro é o *valor* de cada elemento do array e o segundo parâmetro, seu respectivo *índice*. Note que esses dois parâmetros estão em ordem inversa àqueles do método `each()`, estudado em [C2 S2.1].

O parâmetro *inverter* é facultativo e booleano, sendo o valor-padrão `false`. Quando definido para `true`, a função retorna os valores falsos do filtro definido na função.

Para exemplificar, considere um array cujos elmentos são números e aplique nele a função `jQuery.grep()` em quatro etapas sucessivas, com a finalidade de melhor esclarecer seu funcionamento. As etapas sugeridas são as seguintes:

- definir um array de números;
- excluir do array as ocorrências do valor 3;
- no array resultante, excluir as ocorrências entre os índices 2 e 4;
- no array resultante, excluir as ocorrências de valores maiores que 7 e o índice 2.

Exemplo:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('.versao').click(function() {
         var arr = [ 4, 3, 1, 9, 6, 8, 0, 2, 7, 3, 4, 8, 7, 1, 9 ];
         $('#um').append('<p>' + arr.join('- ') + '</p>');
         arr = jQuery.grep(arr, function(val, i) {
            return (val != 3);
         });
         $('#um').append('<p>' + arr.join('- ') + '</p>');
         arr = jQuery.grep(arr, function(val, i) {
            return (i < 2 || i > 4)
         });
         $('#um').append('<p>' + arr.join('- ') + '</p>');
         arr = jQuery.grep(arr, function(val, i) {
```

```
                return (val <= 7 && i != 2)
            });
            $('#um').append('<p>' + arr.join('- ') + '</p>');
        });

        $('.reset').click(function() {
            $('#um').empty();
        });
    });
</script>
</head>
<body>
    <button type="button" class="versao">jQuery.grep()</button>
    <button type="button" class="reset">Reset</button><br />
    <div id="um"></div>
...
```

[arquivo-7.3b.html]

jQuery.makeArray(*objeto*)

Esta função transforma qualquer conjunto de objetos em um array. O método mostrado a seguir coleta o conteúdo de todos os divs do documento e cria um array cujos elementos são esses conteúdos.

```
var arr = jQuery.makeArray('div')
```

É improvável que você precise usar essa função utilitária uma vez que a biblioteca jQuery dela se utiliza nativamente. Lembre-se de que todos os objetos jQuery retornam um conjunto de elementos.

jQuery.map(*array função*(*valor índice*))

Esta função permite mapear os elementos de um array e construir, com base neles, um novo array. O parâmetro `array` define o array a mapear e o parâmetro `função`, uma operação sobre os elementos mapeados com o objetivo de obter os elementos do novo array.

O parâmetro `função` admite dois parâmetros: o primeiro parâmetro é o *valor* de cada elemento do array e o segundo parâmetro, seu respectivo *índice*.

No exemplo a seguir, há um array cujos elementos são letras do alfabeto e a partir dele se construirá um novo array em três etapas, conforme discriminadas a seguir:

- definir um array de letras minúsculas;
- mapear e construir um novo array alterando as letras para maiúsculas e acresentando o índice a cada elemento;
- mapear o array resultante e construir um novo array cujos elementos serão acrescidos de um sinal asterisco (*) imediatamente seguido pela divisão do índice do elemento por cinco.

Exemplo:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('.versao').click(function() {
         var arr = [ 'e', 'g', 'h', 'j', 'v', 'a', 'j', 'm', 'c' ];
         $('div').append('<p>' + arr.join(' - ') + '</p>');

         arr = jQuery.map(arr, function(val, i) {
            return (val.toUpperCase() + i*3);
         });
         $('div').append('<p>' + arr.join(' - ') + '</p>');

         arr = jQuery.map(arr, function(val, i) {
            return (val + '*' + (i/5))
         });
         $('div').append('<p>' + arr.join(' - ') + '</p>');
      });

      $('.reset').click(function() {
         $('div').empty();
      });
   });
</script>
</head>
<body>
   <button type="button" class="versao">jQuery.map()</button>
   <button type="button" class="reset">Reset</button><br />
   <div></div>
...
```

[arquivo-7.3c.html]

jQuery.**inArray(***valor array***)**

Esta função permite inspecionar os elementos de um array e verificar se um determinado elemento a ele pertence.

O parâmetro *valor* é o elemento do array cuja existência se quer verificar e o segundo parâmetro, *array*, é o nome do array a inspecionar.

O valor retornado é o índice da primeira ocorrência do elemento pesquisado, caso seja encontrado no array. Caso não seja encontrado, será retornado o número -1 (um negativo).

Exemplo:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      $('.versao').click(function() {

         var arr = [ 'site', 'maujor', 'h', 2, 'jquery', 105, 'leitor', 'xpto' ];
         $('div').append('<p>' + arr.join(', ') + '</p>');

         var resUm = jQuery.inArray('maujor', arr)
         $('div').append('<p>Procura por maujor resultou índice: <b>' + resUm + '</b></p>');

         var resDois = jQuery.inArray(105, arr)
         $('div').append('<p>Procura por 105 resultou índice: <b>' + resDois + '</b></p>');

         var resTres = jQuery.inArray('LEITOR', arr)
         $('div').append('<p>Procura por LEITOR resultou índice: <b>' + resTres + '</b></p>');

         $('.reset').click(function() {
            $('div').empty();
         });
      });
   });
</script>
</head>
<body>
   <button type="button" class="versao">jQuery.inArray()</button>
   <button type="button" class="reset">Reset</button><br />
   <div></div>
...
```

 [arquivo-7.3d.html]

jQuery.**unique(*array*)**

Esta função remove os elementos que tenham sido anteriormente duplicados (ou clonados) em um array constituído de elementos da árvore do documento e retorna o array sem as duplicatas.

O parâmetro `array` é o array a inspecionar, retirado da árvore do documento. Essa função é de uso restrito e provavelmente você não irá utilizá-la em seus projetos de desenvolvimento de sites. A exemplificação requer o desenvolvimento de marcação e script bastante específicos e, tendo em vista o pouco valor prático retornado em um exemplo, limitamo-nos apenas a citar a função.

7.4 Teste de função

jQuery.**isFunction(objeto)**

Esta função permite inspecionar um objeto qualquer passado no parâmetro *objeto* e retorna `true` se o objeto é uma função, caso contrário retorna `false`.

Exemplo:

```
...
<script type="text/javascript">
  $(document).ready(function() {
    $('.versao').click(function() {

      function maujor() {};
        var objetos = [ maujor, 'site', null, false, 'div', function() {}, 105 ];
        $('div').append('<p>' + objetos.join(', ') + '</p>');

        jQuery.each(objetos, function(i) {
          var resultado = jQuery.isFunction(objetos[i]);
          $('span:eq('+ i +')').html('<b>' + resultado + '</b>');
        });

      $('.reset').click(function() {
        $('div, span').empty();
      });
    });
  });
</script>
</head>
```

```
<body>
    <button type="button" class="versao">jQuery.isFunction()</button>
    <button type="button" class="reset">Reset</button><br />
    <div></div>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
        <span></span>
...
```

[arquivo-7.4a.html]

7.5 Operação com string

jQuery.**trim(***string***)**

Esta função remove os espaços em branco existentes antes e depois de uma string. O parâmetro *string* é a string contendo os espaços a remover.

Exemplo:

```
...
<script type="text/javascript">
    $(document).ready(function() {
        $('button').click(function() {

            var minhaString = "   Foi lançado o livro jQuery do Maujor.        ";
            alert('String antes do trim:\n\n"' + minhaString + '"');

            minhaString = jQuery.trim(minhaString);
            alert('String depois do trim:\n\n"' + minhaString + '"');
        })
    });
</script>
</head>
<body>
    <button type="button">jQuery.trim()</button>
...
```

[arquivo-7.5a.html]

7.6 Funções utilitárias personalizadas

Funções utilitárias personalizadas são aquelas criadas pelo desenvolvedor para suprir necessidades específicas de um determinado projeto ou projetos. Conceitualmente, pode-se dizer que um conjunto de funções utilitárias personalizadas constitui uma biblioteca particular que o desenvolvedor emprega para simplificar a realização de tarefas repetitivas em seu projeto.

Neste item do capítulo, será criada uma função utilitária simples, com o objetivo de mostrar a técnica envolvida no desenvolvimento de tais funções, para o emprego em conjunto com a biblioteca jQuery.

Sabe-se que o *alias* da biblioteca jQuery é `$` e também que várias outras bibliotecas o empregam. Em um desenvolvimento que utilize exclusivamente a biblioteca jQuery, é seguro usar esse *alias* na sintaxe dos métodos, sem prever mecanismos adicionais para prevenir conflitos.

Mesmo que você esteja desenvolvendo seu projeto com exclusividade de uso para a biblioteca jQuery, é de boa prática prevenir conflitos, pois nada garante que futuramente seu projeto venha a ser atualizado ou expandido e você ou outro desenvolvedor tenha que implantar uma outra biblioteca no projeto. Adote como regra não confiar no uso do *alias* `$` sem um mecanismo de prevenção de conflitos.

O primeiro mecanismo é substituir o *alias* `$` por `jQuery`. Por exemplo:

`$.isFunction()` é substituído por `jQuery.isFunction()`.

Em pequenos scripts, essa solução é satisfatória, contudo, em scripts maiores, a grande quantidade de substituições de `$` por `jQuery` acaba por despender tempo, aumentar o trabalho de digitação e complicar o código.

O segundo mecanismo, para não perder a simplicidade do alias `$`, consiste no uso do método `$.noConflict()` estudado em [C2 S2.1].

7.6.1 Sintaxe geral

Observe a função mostrada que se destina a criar uma caixa de alerta contendo o texto passado no parâmetro *`alerta`*:

```
$.alerta = function(alerta) {
   alert(alerta);
}
```

Trata-se de uma função utilitária muito simples, destinada a ser usada com a biblioteca jQuery, mas potencialmente conflitante com outras bibliotecas. Veja a seguir duas formas de chamada para essa função:

```
...
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $.alerta = function(texto) {
         alert(texto);
      }

      $('button:eq(0)').click(function() {
         $.alerta('Alerta com uso de $.alerta');
      });

      $('button:eq(1)').click(function() {
         jQuery.alerta('Alerta com uso de jQuery.alerta');
      });
   });
</script>
</head>
<body>
<button type="button">$.alerta</button>
<button type="button">jQuery.alerta</button>
...
```

[arquivo-7.6.1a.html]

A primeira forma de chamada da função utiliza a sintaxe `$.alerta` e é potencialmente conflitante; a segunda forma é mais segura e evita conflitos usando `jQuery.alerta`. Esse simples exemplo mostra na prática o uso das duas sintaxes de chamada de uma função jQuery.

Mas, como usar o alias `$` sem o risco de conflitos? Seria ótimo se isso fosse possível, não é mesmo? E é possível. A sintaxe JavaScript permite passar `jQuery` como se fosse um parâmetro de substituição ao alias `$` conforme mostrado a seguir:

```
(function($) {
   // Aqui vai o script da função personalizada
})(jQuery);
```

Adotando essa sintaxe para escrever a função, haverá segurança de que o alias `$` não será usurpado por outras bibliotecas.

A função `$.alerta` deverá ser reeescrita assim:

```
(function($) {
   $.alerta = function(textoAlerta) {
      alert(textoAlerta);
   }
})(jQuery);
```

7.6.2 Funçao `$.corTexto`

Para sedimentar os conceitos de criação de funções personalizadas, será desenvolvida como exemplo uma função denominada `$.corTexto`, destinada a mudar a cor do texto de elementos em uma página web.

Em uma primeira etapa, cria-se a função para mudar a cor de textos de parágrafos e, a seguir, aperfeiçoa-se a função para mudar seletivamente a cor de outros elementos da página.

Não há preocupação com o valor prático da função, uma vez que o objetivo é exemplificar conceitos de criação de funções utilitárias.

Observe o código a seguir:

```
1. (function($) {
2.    $.corTexto = function(e, cor) {
3.    var el = document.getElementsByTagName(e);
4.       for (var i = 0; i < el.length; i++) {
5.          el[i].style.color = cor;
6.       }
7.    }
8. })(jQuery);
```

Código comentado:

Linha(s)	Descrição
Linhas 1 e 8	A função está contida dentro de `function($)`, destacada em negrito, conforme a sintaxe estudada no item anterior.
Linha 2	A função admite o parâmetro `e`, que define o elemento, e o parâmetro `cor`, que define a cor a ser adotada para o elemento.
Linha 3	Cria-se uma variável denominada `el` que armazena, em um array, todos os elementos `e` existentes na página.
Linha 4	Cria-se um loop para percorrer os elementos armazenados na variável `el`.
Linha 5	Aplica-se uma regra de estilo definindo a cor passada no parâmetro *cor* para todos os elementos `el`.

Tal função está pronta para uso, conforme mostrada a seguir:

```
...
<script type="text/javascript">
   $(document).ready(function() {
      (function($) {
         $.corTexto = function(e, cor) {
         var el = document.getElementsByTagName(e);
            for (var i = 0; i < el.length; i++) {
               el[i].style.color = cor;
            }
         }
      })(jQuery);

      $('button:eq(0)').click(function() {
         $.corTexto('p', '#ff0000');
      });
      $('button:eq(1)').click(function() {
         jQuery.corTexto('p', '#0000ff');
      });
      $('button:eq(2)').click(function() {
         $.corTexto('p', '#00cc00');
      });
      $('button:eq(3)').click(function() {
         jQuery.corTexto('p', '');
      });
   });
</script>
</head>
<body>
   <span>Escolha uma cor para os parágrafos:</span><br />
   <button type="button">Vermelha</button>
   <button type="button">Azul</button>
   <button type="button">Verde</button>
   <button type="button">Default</button>
<p>Parágrafo um</p>
<p> Parágrafo dois</p>
<p> Parágrafo três</p>
...
```

[arquivo-7.6.2a.html]

Tendo entendido e consultado o arquivo que demonstra o funcionamento dessa função, talvez você esteja se questionando do porquê de todo este trabalho se se pode usar simplesmente `$('p').css('color', '#xxxxxx')` com um valor `#xxxxxx` para cada um dos botões de mudança de cor?

É verdade. Se a função terminasse aqui, não teria valor prático algum, pois se limitou a replicar com mais código um método nativo. E funções utilitárias não se destinam a substituir métodos nativos, e sim complementá-los ou suprir suas faltas.

Assim sendo, adicione certa flexibilidade a essa função, justificando sua criação, ainda que de uso limitado, pois como se disse, o objetivo é mostrar a sintaxe e fundamentos de criação de funções nativas.

Que tal se em vez de clicar um botão para mudar a cor dos parágrafos, você tivesse um menu de seleção de cores de modo que toda vez que o usuário selecionasse uma cor, houvese mudança sem necessidade de cliques?

Retire os botões da página e coloque um menu de opções de cores, conforme mostrado a seguir:

```
<form action="" method="">
    <label>Escolha uma cor para os parágrafos<br />
    <select id="selecionaCor">
        <option value="#999999">Default</option>
        <option value="#FF0000">Vermelha</option>
        <option value="#0000FF">Azul</option>
        <option value="#00CC00">Verde</option>
    </select>
    </label>
</form>
```

Não há muito o que comentar sobre a marcação do menu de seleção. Note a definição do `id="selecionaCor"` para o elemento `select` e a opção de cor default `#999999`, supondo que esta é a cor-padrão adotada para os parágrafos da página e definida por regra CSS.

Nessa nova solução, a cor será coletada do menu de seleção e não mais passada como um parâmetro da função, como era na solução anterior. Assim, tal função terá apenas o parâmetro `el` e será necessário informar qual foi a cor escolhida pelo usuário no próprio corpo da função.

Observe as modificações introduzidas na função:

```
1. (function($) {
2.     $.corTexto = function(e) {
3.         var selecao = document.getElementById('selecionaCor');
4.         var indiceSelecao = selecao.selectedIndex;
5.         sel = selecao.options[indiceSelecao].value;
6.         var el = document.getElementsByTagName(e);
```

```
7.         for (var i = 0; i < el.length; i++) {
8.                el[i].style.color = sel;
9.            }
10.        }
11. })(jQuery);
```

Código comentado:

Linha(s)	Descrição
Linha 2	A função requer apenas um parâmetro, já que a cor será passada pelo menu de seleção.
Linha 3	Cria-se uma variável denominada `selecao` que armazena o elemento `select` com `id="selecionaCor"`.
Linha 4	Recolhe-se o índice da cor selecionada na variável `indiceSelecao`.
Linha 5	Armazena-se a cor escolhida na variável `sel`.
Linha 8	Informa-se que a cor a ser aplicada é aquela armazenada na variável `sel`, que, por sua vez, foi a escolhida pelo usuário.

Nessa nova solução, tão logo o usuário escolha a cor, esta será automaticamente aplicada aos parágrafos, sem necessidade de clicar um botão.

A seguir, o script completo dessa solução:

```
...
<style type="text/css" media="all">
   p {color:#999;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      (function($) {
         $.corTexto = function(e) {
            var selecao = document.getElementById('selecionaCor');
            var indiceSelecao = selecao.selectedIndex;
            sel = selecao.options[indiceSelecao].value;
            var el = document.getElementsByTagName(e);
            for (var i = 0; i < el.length; i++) {
               el[i].style.color = sel;
            }
         }
      })(jQuery);
      $('#selecionaCor').change(function() {
         $.corTexto('p');
      });
   });
</script>
</head>
```

```
<body>
<form action="" method="">
    <label>Escolha uma cor para os parágrafos<br />
    <select id="selecionaCor">
        <option value="#999999">Default</option>
        <option value="#FF0000">Vermelha</option>
        <option value="#0000FF">Azul</option>
        <option value="#00CC00">Verde</option>
    </select></label>
</form>
<p>Parágrafo um</p>
<p>Parágrafo dois</p>
<p>Parágrafo três</p>
...
```

[arquivo-7.6.2b.html]

Você seria capaz de dizer se os resultados obtidos com essa solução podem ser conseguidos com métodos nativos do jQuery? Tal desafio foi lançado como exercício prático. Encontrou uma solução? Poste na área de comentários do site do livro.

Que tal incrementar mais um pouco essa função permitindo que o usuário escolha, além da cor, os elementos que devam ser afetados pela escolha?

As modificações a introduzir no script são as descritas a seguir.

- Criar botões para selecionar cada elemento a mudar de cor e um botão para selecionar todos os elementos a mudar de cor:

```
<button type="button">Parágrafos</button>
<button type="button">Cabeçalhos</button>
<button type="button">Listas</button>
<button type="button">Tudo</button>
```

- Criar nova marcação para a página que contém os elementos a mudar de cor:

```
<h2>Cabeçalho nível 2</h2>
    <p>Parágrafo um</p>
    <p>Parágrafo dois</p>
    <p>Parágrafo três</p>
<h2>Cabeçalho nível 2</h2>
    <ul>
        <li>Item um</li>
        <li>Item dois</li>
        <li>Item três</li>
    </ul>
    <p>Parágrafo</p>
<h2>Cabeçalho nível 2</h2>
```

- Retirar o disparador de mudança de cor do elemento `select` e atribuí-lo aos botões:

```
$('button:eq(0)').click(function() {
    $.corTexto('p');
});
$('button:eq(1)').click(function() {
    jQuery.corTexto('h2');
});
$('button:eq(2)').click(function() {
    $.corTexto('li');
});
    $('button:eq(3)').click(function() {
jQuery.corTexto('*');
});
```

Com essas modificações, chega-se a uma terceira solução para a função personalizada cujo funcionamento pode ser visto no arquivo a seguir.

 [arquivo-7.6.2c.html]

Modifique tal função criando outro elemento `select` para permitir ao usuário escolher o elemento a mudar de cor, tal como se fez para a escolha da cor. Esse novo `select` substituirá os botões de seleção dos elementos. A técnica é idêntica à adotada anteriormente para o `select` de cores e a modificação em relação ao script anterior é mostrada a seguir:

```
...
<script type="text/javascript">
    $(document).ready(function() {
        (function($) {
            $.corTexto = function() {
                var selecaoCor = document.getElementById('selecionaCor');
                var indiceSelecaoCor = selecaoCor.selectedIndex;
                var selCor = selecaoCor.options[indiceSelecaoCor].value;
                var selecaoElemento = document.getElementById('selecionaElemento');
                var indiceSelecaoElemento = selecaoElemento.selectedIndex;
                var selElemento = selecaoElemento.options[indiceSelecaoElemento].value;
                var el = document.getElementsByTagName(selElemento);
                for (var i = 0; i < el.length; i++) {
                    el[i].style.color = selCor;
                }
            }
        })(jQuery);
```

```
        $('button').click(function() {
            $.corTexto();
        });
    });
...
```

[arquivo-7.6.2d.html]

7.6.2.1 Melhorando a função `$.corTexto`

Como você deve ter acompanhado no desenvolvimento da função personalizada `$.corTexto`, extraiu-se o valor que o usuário entrou no campo de seleção para a cor, em três etapas distintas a saber:

- uso do método `document.getElementById` para selecionar o elemento `select`;
- uso de `selectIndex` para capturar o índice da seleção;
- uso de `value` para capturar o valor selecionado pelo usuário.

Para a seleção do elemento, a técnica usada foi idêntica.

Uma análise atenta da solução proposta mostra que esta é falha e deve ser melhorada. Note que se utilizou um método JavaScript sem testar seu suporte pelo navegador. As boas práticas de programação recomendam a seguinte verificação para o método usado:

```
if (!document.getElementById) {
    return false;
}
```

Isto assegura que o navegador não tentará executar o código caso não ofereça suporte para o método. Além disso, continua-se tal desenvolvimento usando métodos JavaScript não específicos à biblioteca jQuery.

Realizou-se tal ação propositadamente, pois é comum com desenvolvedores familiarizados com JavaScript cometer o equívoco de inserir métodos tradicionalmente usados naquela programação, sem se preocupar com a portabiliade e o suporte adequado para o método. Quanto mais familiarizado estiver com a biblioteca, tanto mais fácil será encontrar a alternativa para o código JavaScript não jQuery. Isto não significa que você está proibido de usar a sintaxe da linguagem JavaScript em seus desenvolvimentos com a biblioteca jQuery, ao contrário, use sempre que necessário, mas assegure-se de não estar poluindo seus scripts com métodos em desacordo com a filosofia da biblioteca.

Você se lembra do seletor `:selected` estudado em [C2 S2.2.9]? e do método `val()` estudado em [C3 S3.5]? Pois bem, essas duas funcionalidades da biblioteca, além de compactar o código da função personalizada, oferecem uma solução bem mais robusta e em conformidade com os fundamentos de jQuery, sem a necessidade de verificar compatibilidades.

Veja a seguir a alternativa para a segunda etapa de desenvolvimento da função. O arquivo `arquivo-7.6.2b.html` que representa tal etapa foi reescrito como mostrado a seguir:

```
...
<style type="text/css" media="all">
   p {color:#999;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('select').change(function() {
         $('option:selected').each(function() {
            $('p').css('color', $(this).val());
         });
      });
   });
</script>
</head>
<body>
<form action="" method="">
   <label>Escolha uma cor para os parágrafos<br />
   <select id="selecionaCor">
      <option value="#999999">Default</option>
      <option value="#FF0000">Vermelha</option>
      <option value="#0000FF">Azul</option>
      <option value="#00CC00">Verde</option>
   </select>
   </label>
</form>
<p>Parágrafo um</p>
<p>Parágrafo dois</p>
<p>Parágrafo três</p>
...
```

[arquivo-7.6.2.1a.html]

E, a seguir, a alternativa para a terceira etapa de desenvolvimento da função. O arquivo `arquivo-7.6.2c.html` que representa tal etapa foi reescrito como mostrado a seguir:

```
...
<style type="text/css" media="all">
   * {color:#999;}
   p {margin:2px 0;}
   form {display:inline;}
   select {width:105px;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      (function($) {
         $('select').change(function() {
            $('option:selected').each(function() {
            var cor = $(this).val();
               $.corTexto = function(e) {
               $(e).css('color', cor);
               }
            });
         });
      })(jQuery);
      $('button:eq(0)').click(function() {
         $.corTexto('p');
      });
      $('button:eq(1)').click(function() {
         jQuery.corTexto('h2');
      });
      $('button:eq(2)').click(function() {
         $.corTexto('li');
      });
      $('button:eq(3)').click(function() {
         jQuery.corTexto('*');
      });
   });
</script>
</head>
<body>
<form action=" " method=" ">
   <label>Escolha uma cor<br />
   <select id="selecionaCor">
      <option value="#999999">Default</option>
      <option value="#FF0000">Vermelha</option>
```

```
            <option value="#0000FF">Azul</option>
            <option value="#00CC00">Verde</option>
            <option value="#FF00FF">Pink</option>
        </select>
        </label>
    </form>
    <button type="button">Parágrafos</button>
    <button type="button">Cabeçalhos</button>
    <button type="button">Listas</button>
    <button type="button">Tudo</button>
    <h2>Cabeçalho nível 2</h2>
    <p>Parágrafo um</p>
    <p>Parágrafo dois</p>
    <p>Parágrafo três</p>
    <h2>Cabeçalho nível 2</h2>
    <ul>
    <li>Item um</li>
    <li>Item dois</li>
    <li>Item três</li>
    </ul>
    <p>Parágrafo</p>
    <h2>Cabeçalho nível 2</h2>
    ...
```

[arquivo-7.6.2.1b.html]

A função pode ser aperfeiçoada ainda mais e ganhar funcionalidades adicionais, mas tal exemplo será finalizado aqui, pois já cumpriu o objetivo de mostrar as técnicas de criação de funções personalizadas. Ressalta-se que a função criada manipula elementos do DOM, mas como já se disse, é possível manipular objetos não pertencentes ao DOM com funções personalizadas. Que tal desenvolver uma função capaz de manipular o objeto `Date()` e personalizar a apresentação de datas em uma página web?

Parte II

jQuery na prática

A partir do capítulo 8, inicia-se a segunda parte deste livro que é dedicada a aplicação prática dos conceitos e fundamentos estudados na primeira parte. Todos os exemplos desenvolvidos são comentados e estão disponíveis para download no site do livro. Recomenda-se que o estudo dos casos mostrados seja sempre complementado com a visualização do respectivo arquivo.

CAPÍTULO 8

Animações básicas

Neste capítulo, serão apresentadas algumas animações básicas da biblioteca jQuery. Para cada exemplo, serão mostradas e comentadas a marcação HTML e as regras CSS aplicadas, pois é importante que a aplicação de qualquer efeito seja feita em documentos estruturados em conformidade com os padrões do W3C. Separar a marcação HTML da apresentação é igualmente importante e assim se enfatizarão, também, as regras CSS, comentando-as sempre que se julgar necessário.

8.1 Marcação mínima

Todos os exemplos desenvolvidos não só neste capítulo, mas também nos capítulos subsequentes, se utilizam de uma marcação estrutural mínima conforme se mostra a seguir:

```
1.     <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
2.     Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3.     <html xmlns="http://www.w3.org/1999/xhtml" lang="pt-br" xml:lang="pt-br">
4.     <head>
5.     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
6.     <title>Exemplos práticos</title>
7.     <style type="text/css" media="all">
8.        /* Regras de estilo aqui */
9.     </style>
10.    <script type="text/javascript" src="../jquery-1.2.6.js"></script>
11.    <script type="text/javascript">
12.       $(document).ready(function() {
13.          // scripts jQuery devem ser inseridos aqui
14.       });
15.    </script>
```

```
16.     </head>
17.     <body>
18.         <!-- Conteúdo da página aqui -->
19.     </body>
20.     </html>
```

 [arquivo-8.1a.html]

Código comentado:

Linha(s)	Descrição
Linhas 1 e 2	Todo documento HTML deve começar com a declaração do DOCTYPE. É nessa declaração que o navegador busca as informações sobre a sintaxe da marcação, bem como instruções de como renderizar cada elemento da marcação. A maioria dos documentos existentes omite essa declaração e a renderização se processa no chamado modo de renderização kirks, em oposição ao documento corretamente marcado cujo modo de renderização é chamado de standard. Uma das mais notáveis inconsistências em consequência da falta de DOCTYPE em um documento é o "Box Model quebrado" no navegador Internet Explorer. Veja `jQuery.boxModel` em [C7 S7.2].
Linha 3	Segue-se ao DOCTYPE a tag de abertura do elemento `html` que é o elemento-raiz do documento. Use o atributo `lang` para indicar o idioma principal do documento.
Linha 5	Sempre declare a codificação de caracteres do documento.
Linha 6	Não omita o título de seu documento. Use títulos que informem o conteúdo da página, pois os mecanismos de busca dão muita importância ao conteúdo do elemento `title`. O título que se utilizou "Exemplos práticos" tem caráter puramente didático. Na prática, é uma péssima escolha. Melhor seria descrever sumariamente quais são os tipos de exemplo contidos na página, por exemplo: exemplos dos métodos jQuery `show()` e `hide()`.
Linhas 7 a 9	A folha de estilo do documento será escrita dentro do elemento `style`. Serão usadas folhas de estilo incorporadas ao documento para facilitar a leitura e o estudo. Folhas de estilo, salvo exceções, devem ser colocadas em arquivos externos e linkadas ao documento.
Linha 10	Aqui se linkou para a biblioteca jQuery gravada em um arquivo externo chamado `jquery-1.2.6.js`, que é a última versão da biblioteca à época que este livro foi escrito e a qual foi baixada em `http://docs.jquery.com/Downloading_jQuery`.
Linhas 11 a 15	Contido dentro do elemento script, inseriu-se o método jQuery `$(document).ready()` que é o equivalente ao método JavaScript `window.onload`. Todo script ali contido só entra em ação depois que o DOM estiver pronto, ou seja, depois que o documento tiver sido carregado.

Linha(s)	Descrição (cont.)
Linhas 17 a 19	Completam o documento marcando seus conteúdos. Tudo que deve ser renderizado será incluído na seção `body`.
Linha 20	Fechamento do elemento-raiz do documento.

Para maiores informações sobre DOCTYPEs padrão do W3C, consulte: `http://www.w3.org/QA/2002/04/valid-dtd-list.html`.

8.2 Animação com `show()` e `hide()`

Show e hide são palavras inglesas que significam, respectivamente, mostrar e esconder. Os métodos jQuery `show()` e `hide()` se destinam a mostrar e esconder elementos do DOM. Observe as regras CSS a seguir:

```
1.    p.esconde {display:none;}
2.    p.mostra {display:block;}
```

Código comentado:

Linha(s)	Descrição
Linha 1	Esta regra CSS se emprega nos parágrafos em que se aplicou a classe `esconde`. A declaração `display:none` faz que os conteúdos desses parágrafos não sejam renderizados e tudo se passa como se tais conteúdos não constassem da marcação. Os conteúdos são retirados do fluxo normal do documento e o espaço físico por eles ocupado na renderização é preenchido pelo elemento que se segue no fluxo da marcação. Convém ressaltar que elementos com `display:none` são ignorados também por tecnologias assistivas, como leitores de tela, e assim permanecem inacessíveis tanto visualmente como para qualquer tipo de mídia.
Linha 2	Esta regra CSS se emprega nos parágrafos em que se aplicou a classe `mostra`. A declaração `display:block` faz que os conteúdos desses parágrafos sejam renderizados normalmente. Esta declaração é a declaração-padrão para parágrafos e todos os demais elementos nível de bloco da marcação HTML.

Para elementos in-line, tais como `span`, `strong`, `em`, `img` etc., a condição de visibilidade-padrão é determinada pela regra CSS `display:inline`.

Os métodos `show()` e `hide()` exercem o mesmo efeito que as regras CSS mostradas. Observe os códigos a seguir:

```
1.    $('p.esconde').hide();
2.    $('p.mostra').show();
```

Código comentado:

Linha(s)	Descrição
Linha 1	Este encadeamento jQuery destina-se a encontrar todos os parágrafos aos quais se tenha atribuído a classe `esconde` e a retirá-los do processo de renderização tal como se tivesse sido declarada para eles a regra CSS `display:none`.
Linha 2	Este encadeamento jQuery destina-se a encontrar todos os parágrafos aos quais se tenha atribuído a classe `mostra` e a renderizá-los normalmente tal como se tivesse sido declarada para eles a regra CSS `display:block`.

Criou-se uma página web simples para exemplificar os efeitos que serão estudados neste capítulo. A página inicia-se com um cabeçalho nível 1 e cinco parágrafos, sendo atribuída a dois deles a classe `esconde`. Esta página servirá de laboratório para os scripts de exemplo. Serão explicados cada componente da marcação, bem como a folha de estilo a esta incorporada. À medida que for necessário acrescentar ou suprimir conteúdos na página, para demonstrar um novo efeito, este será acrescentado e será explicada sua necessidade.

A marcação inicial é a seguinte:

```
...
1.     <div id="com-classe">
2.        <label>Parágrafos com classe "esconde"<br /></label>
3.        <button type="button">Esconder</button>
4.        <button type="button">Mostrar</button>
5.     </div>
6.     <div id="todos">
7.        <label>Todos os parágrafos<br /></label>
8.        <button type="button">Esconder</button>
9.        <button type="button">Mostrar</button>
10.    </div>
11.    <h1>Folhas de estilo em cascata</h1>
12.       <p>CSS é a sigla em ...</p>
13.       <p> A grande vantagem ...</p>
14.       <p>HTML marca e ...</p>
15.       <p class="esconde">A adoção desta ...</p>
16.       <p class="esconde">Projetar um ...</p>
...
```

Código comentado:

Linha(s)	Descrição
Linhas 1 a 5	Um `div` com seu respectivo `id` serve de container para os dois botões que se destinam a disparar os eventos esconder e mostrar os parágrafos aos quais se tenha atribuído uma classe `esconde`.

O documento está pronto para receber o script jQuery que fará esconder e mostrar parágrafos. O script é apresentado a seguir:

```
1.    <script type="text/javascript" src="../jquery-1.2.6.js"></script>
2.    <script type="text/javascript">
3.       $(document).ready(function() {
4.          $('button:eq(0)').click(function() {
5.             $('p.esconde').hide();
6.          });
7.          $('button:eq(1)').click(function() {
8.             $('p.esconde').show();
9.          });
10.         $('button:eq(2)').click(function() {
11.            $('p').hide();
12.         });
13.         $('button:eq(3)').click(function() {
14.            $('p').show();
15.         });
16.      });
17.   </script>
```

Código comentado:

Linha(s)	Descrição
Linha 1	Linka a biblioteca jQuery no documento.
Linhas 2 e 17	Abre e fecha o container `<script></script>` para o código.
Linhas 3 e 16	Abre e fecha o container para o método `document.ready()` que faz que todo o script nele contido aguarde o carregamento do DOM para entrar em ação.
Linha 4	Define uma função a ser executada quando o usuário clicar o primeiro elemento `button` na página. Lembre-se de que os métodos jQuery retornam um conjunto de objetos, sem necessidade de se proceder a loops. Assim `$('button')` retorna todos os botões encontrados no documento e a estes atribui um índice começando a contagem em 0 (zero), logo `eq(0)` se refere ao primeiro elemento `button` encontrado.
Linha 5	Este é o script da função que será executada ao se clicar o primeiro botão. Encontra todos os parágrafos com a classe `esconde`, e esconde-os, caso estejam visíveis.
Linhas 7 a 9	Este é o script da função que será executada ao se clicar o segundo botão. Encontra todos os parágrafos com a classe `esconde` e mostra-os, caso estejam escondidos.
Linhas 10 a 15	Cumprem funções idênticas às anteriores para todos os parágrafos do documento.

Para a demonstração desse efeito, consulte o arquivo indicado a seguir.

Conforme mostra a figura 8.1, a página construída para as demonstrações possui dois conjuntos de botões, um à esquerda e outro à direita. Cada um desses conjuntos cumpre finalidades distintas na demonstração. Ao visualizar os efeitos em seu navegador, clique sucessivamente o conjunto de botões à esquerda e, depois, o conjunto de botões à direita. Observe e entenda o funcionamento de cada um dos conjuntos separadamente, a partir da posição inicial da página, ou seja, comece os cliques em cada conjunto com todos os cinco parágrafos à vista.

 [arquivo-8.2a.html]

O script está funcionando como se pretendeu, mas há algumas melhorias que se pode introduzir. O método jQuery `toggle()` destina-se a alternar as condições de um elemento do documento. Em se tratando de visibilidade, se o elemento estiver visível, ocultará, e se estiver oculto, acabará tornando-o visível.

8.3 Animação com `toggle()`

O uso deste método simplifica não só o script, mas também a marcação da página. Possibilita que se use um único botão para cumprir duas finalidades, no caso em questão, mostrar e esconder.

Genericamente, o método `toggle()` permite alternar dois comportamentos, e não somente a visibilidade, a cada clique do mouse. Você define dois comportamentos como parâmetros para o método `toggle()` e quando o usuário clica uma vez, dispara o primeiro comportamento definido, outro clique, o segundo comportamento, mais um clique, o primeiro comportamento, e assim indefinidamente vão se alternando os comportamentos definidos conforme o clique seja par ou ímpar. Observe que no script a seguir, definiu-se o método `toggle()` aplicado a um `div`. A primeira função reduz a opacidade do `div` para 20% e a segunda função restitui a opacidade original para 100%, ou seja, opaca.

As regras CSS para apresentar o `div` são mostradas a seguir.

CSS:

```
...
<style type="text/css" media="all">
div {
   width:200px;
   height:150px;
   background:#ff0;
   border:2px solid #000;
   cursor:pointer;
   }
</style>
```

Agora você precisa somente de um botão.

Código comentado:

Linha(s)	Descrição
Linha 4	Encontre todos os parágrafos com a classe `esconde` e torne-os invisíveis, caso estejam visíveis, ou visíveis, caso estejam invisíveis.
Linha 7	Cumpre as mesmas funções descritas anteriormente para todos os parágrafos do documento.

Para a demonstração desse efeito, com o uso de `toggle()`, consulte o arquivo indicado e mostrado na figura 8.2.

Figura 8.2 – Solução com um botão e dois textos.

No arquivo indicado, observe o funcionamento dos scripts considerando que ambos agem sobre os mesmos parágrafos, ou seja, ao utilizar `toggle()`, o resultado do último clique no botão à esquerda influenciará o primeiro clique no botão à direita e vice-versa.

Para evitar confusão, verifique o funcionamento do botão à esquerda realizando vários cliques seguidos nele e termine com o clique que mostra todos os parágrafos. A seguir, aplique o mesmo procedimento ao botão à direita.

A seguir, alterne os cliques nos botões e observe os resultados.

Agora, dê outro passo no sentido de aperfeiçoar ainda mais esse documento alterando a maneira como o texto do botão é apresentado ao usuário. Que tal se em vez do texto Esconder/Mostrar para os botões, houvesse o texto "Esconder" se os parágrafos estivessem à mostra e "Mostrar" se estivessem escondidos? Para obter esse aperfeiçoamento, retire o texto dos botões da marcação HTML conforme mostrado a seguir:

```
...
<button type="button"></button>
...
```

E acrescente ao script anterior o seguinte:

```
1.    <script type="text/javascript">
2.       $(document).ready(function() {
3.          $('button:eq(0)').text('Esconder').click(function() {
4.             if ($(this).text() == 'Mostrar') {
5.                $(this).text('Esconder');
6.             } else {
7.                $(this).text('Mostrar');
8.             }
9.             $('p.esconde').toggle();
10.         });
11.         $('button:eq(1)').text('Esconder').click(function() {
12.            if ($(this).text() == 'Mostrar') {
13.               $(this).text('Esconder');
14.            } else {
15.               $(this).text('Mostrar');
16.            }
17.            $('p').toggle();
18.         });
19.      });
20.   </script>
```

Código comentado:

Linha(s)	Descrição
Linha 3	Utilizou-se o método `text('Esconder')`, que se destina a inserir o texto "Esconder", como conteúdo do botão. Assim, quando o documento é carregado, o botão aparece com esse texto.
Linha 12	Ao ser clicado, inicia-se um teste condicional que determina o seguinte: se o elemento que acaba de ser clicado, `$(this)`, contiver o texto "Mostrar", altere esse texto para "Esconder", se não, altere para "Mostrar". Esse teste condicional faz a mágica, alternando o texto do botão.

Para demonstrar esse novo efeito, com alternância do texto do botão, consulte o arquivo indicado e mostrado na figura 8.3.

Figura 8.6 – Solução com CSS desabilitada.

Note que os botões são renderizados, mas perderam o respectivo texto indicativo de sua finalidade. É necessário corrigir isso. Você saberia antecipar a solução para esse novo problema? Ou seja, como apresentar os botões completos na situação em que JavaScript está habilitado e CSS está desabilitada?

A resposta é a seguinte: retire a marcação dos botões do HTML e insira-os via jQuery. E aqui se usarão os métodos `before()` e `after()` que inserem marcação, respectivamente, antes e depois de um elemento do DOM.

O script será alterado para o seguinte:

```
1.    <script type="text/javascript">
2.        $(document).ready(function() {
3.            var htmlBotaoUm = "<div id='com-classe'><label>Parágrafos com classe
                  'esconde'<br /></label><button type='button'></button></div>";
4.            var htmlBotaoDois = '<div id="todos"><label>Todos os parágrafos
                  <br /></label><button type="button"></button></div>';

5.            $('h1').before(htmlBotaoUm);
6.            $('#com-classe').after(htmlBotaoDois);
```

```
7.          $('button:eq(0)').text('Esconder').click(function() {
8.             if ($(this).text() == 'Mostrar') {
...               // restante do script sem alteração
23.    });
```

[arquivo-8.3e.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Criou-se a variável `htmlBotaoUm` que armazena a marcação HTML do primeiro botão.
Linha 4	Criou-se a variável `htmlBotaoDois` que armazena a marcação HTML do segundo botão.
Linha 5	Inseriu-se a marcação do primeiro botão imediatamente antes do elemento `h1`.
Linha 6	Inseriu-se a marcação do segundo botão imediatamente após a marcação do primeiro botão que fora inserido conforme descrito na linha 5.

Para demonstrar esse novo efeito, que contempla o caso de CSS desabilitada, consulte o arquivo indicado e mostrado na figura 8.7.

Figura 8.7 – Solução final com CSS desabilitada.

8.4 Suavizando a animação

Você deve ter notado nos arquivos de demonstração desenvolvidos até aqui que a transição mostra/esconde é feita de maneira abrupta, ou seja, os conteúdos são revelados ou escondidos instantaneamente, aparecendo e desaparecendo em frações de segundo.

A biblioteca jQuery oferece uma funcionalidade que permite ao desenvolvedor controlar o intervalo de tempo gasto para mostrar e/ou esconder os conteúdos. Você passa um parâmetro para definir um intervalo de tempo para um dos seguintes métodos: `hide()`, `show()` ou `toggle()`, conseguindo assim suavizar sua animação com estes.

Os parâmetros aceitos são `slow`, `normal` e `fast`, que definem, respectivamente, animações lentas, normais e rápidas. Podemos definir o parâmetro usando unidade de tempo milissegundos, por exemplo: 3000 equivale a um intervalo de 3 segundos, 600 equivale a 0,6 segundo, e assim por diante. Observe os exemplos a seguir:

```
hide('slow');
show('fast');
hide(1200);
toggle(2000);
```

Note o uso das aspas (simples ou duplas) quando a definição do intervalo de tempo é por palavra-chave e o não uso das aspas, quando por milissegundos.

Retome o último arquivo de demonstração: `arquivo-8.3e.html` e suavize a animação aí inserida. Observe os acréscimos feitos:

```
...
$('button:eq(0)').text('Esconder').click(function() {
    if ($(this).text() == 'Mostrar') {
        $(this).text('Esconder');
    } else {
        $(this).text('Mostrar');
    }
        $('p.esconde').toggle('slow');
});

$('button:eq(1)').text('Esconder').click(function() {
    if ($(this).text() == 'Mostrar') {
        $(this).text('Esconder');
```

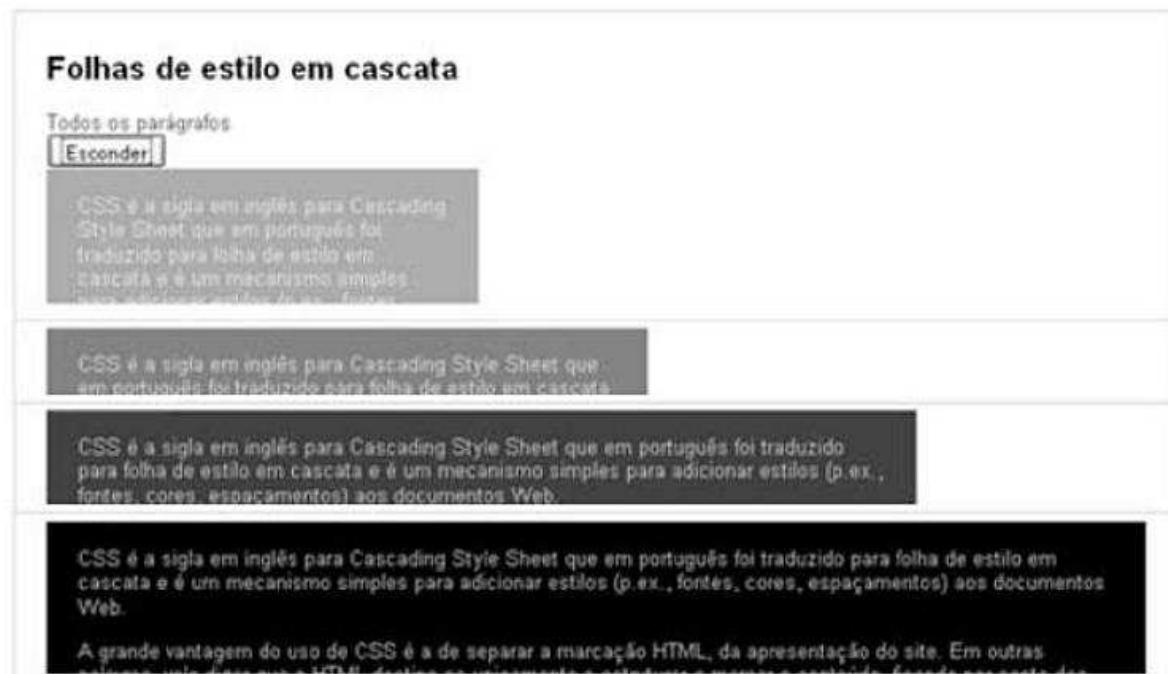



Figura 8.8 – Progressão da animação com show().

Observe os códigos mostrados a seguir:

```
...
<style type="text/css" media="all">
   body {/* sem alterações */}
   h1 {font-size:1.6em;}
   label {color:#c30;}
   div#tudo {background:#000; color:#0f0; padding:2px 20px;}
   button {/* sem alterações */}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      var htmlBotaoUm = '<div id="todos"><label>Todos os parágrafos<br /></label>
         <button type="button"></button></div>';
      $('h1').after(htmlBotaoUm);
      $('button').text('Esconder').click(function() {
         if ($(this).text() == 'Mostrar') {
            $(this).text('Esconder');
            $('#tudo').show(3000);
         } else {
            $(this).text('Mostrar');
            $('#tudo').hide(1000);
         }
      });
   });
</script>
</head>
```

```
<body>
<h1>Folhas de estilo em cascata</h1>
    <div id="tudo">
        <p>CSS é a sigla em ...</p>
        <p> A grande vantagem ...</p>
        <p>HTML marca e estrutura ...</p>
        <p>A adoção desta ...</p>
        <p class="esconde">Projetar um ...<p>
    </div>
...
```

[arquivo-8.4c.html]

8.5 Animação com `slideUp()` e `slideDown()`

Slide up e slide down são palavras inglesas que significam, respectivamente, deslizar para cima e deslizar para baixo. Os métodos jQuery `slideUp()` e `slideDown()` destinam-se a esconder e mostrar, respectivamente, elementos do DOM fazendo que o desaparecimento se processe em um deslizamento de baixo para cima e o aparecimento de cima para baixo, ao contrário dos métodos estudados anteriormente, nos quais o processo mostrar e esconder ocorre em sentido diagonal. Nessa animação, não há efeito de opacidade como na anterior. Na figura 8.9, veja o progresso da animação.

Figura 8.9 – Progressão da animação com `slideDown()`.

Você pode controlar o intervalo de tempo da animação declarando um parâmetro para o método, tal como fez para os métodos estudados anteriormente.

Aplique esse método na animação vista no `arquivo-8.4c.html`. As modificações nesse arquivo são mostradas a seguir:

```
...
$('button').text('Esconder').click(function() {
    if ($(this).text() == 'Mostrar') {
        $(this).text('Esconder');
        $('#tudo').slideDown('slow');
    } else {
        $(this).text('Mostrar');
        $('#tudo').slideUp(1500);
    }
});
...
```

[arquivo-8.5a.html]

Uma funcionalidade interessante da biblioteca jQuery, muito útil para animações, são as chamadas opcionais de uma função denominadas `callback`. Sem outras considerações teóricas, `callback` para animações pode ser definida como uma chamada para uma função que será executada tão logo a animação termine.

Todos os métodos que você coloca em uma animação são executados simultaneamente, respeitado o tempo de execução, resultando na animação.

Caso queira incluir uma ação a ser executada tão logo a animação termine, você deverá passar uma função para o método da animação, como se fora um parâmetro. A sintaxe geral para `callback` é mostrada na animação seguinte, que é uma modificação da animação vista no exemplo anterior. Note no código a seguir que `callback` é uma função que, ao terminar a animação definida com `slideDown()` (revelar), executa a ação de adicionar uma regra CSS definindo uma cor vermelha para os textos contidos e uma cor vermelha para a borda do elemento `body`. Da mesma forma, a função a executar ao término da animação definida com `slideUp()` (esconder) executa uma ação de mudança de cores.

```
...
$('button').text('Esconder').click(function() {
    if ($(this).text() == 'Mostrar') {
        $(this).text('Esconder');
            $('#tudo').slideDown(3000, function() {
                $('body').css({color:'red', borderColor:'red'});
            });
```

```
    } else {
        $(this).text('Mostrar');
            $('#tudo').slideUp(1500, function() {
                $('body').css({color:'green', borderColor:'green'});
            });
        };
...
```

[arquivo-8.5b.html]

8.6 Animação com `fadeIn()` e `fadeOut()`

Fade out e fade in são palavras inglesas que significam, respectivamente, desvanecer e avivar. Os métodos jQuery `fadeOut()` e `fadeIn()` se destinam, respectivamente, a esconder e a mostrar elementos do DOM fazendo que o desaparecimento se processe com o esmaecimento do conteúdo e o aparecimento, com seu avivamento. Funcionam de maneira diferente dos métodos estudados anteriormente, nos quais o processo mostrar e esconder se faz com movimentação do conteúdo, quer na diagonal, quer na vertical. Nesta animação, o efeito é obtido exclusivamente com variação de opacidade e o conteúdo não se movimenta. Na figura 8.10, veja o progresso da animação.

Figura 8.10 – Progressão da animação com `fadeIn()`.

Você pode controlar o intervalo de tempo da animação declarando um parâmetro para o método, tal como fez para os métodos estudados anteriormente.

Aplique esse método na animação vista no `arquivo-8.5a.html`. As modificações nesse arquivo são mostradas a seguir:

```
...
$('button').text('Esconder').click(function() {
    if ($(this).text() == 'Mostrar') {
        $(this).text('Esconder');
        $('#tudo').fadeIn(5000);
    } else {
        $(this).text('Mostrar');
        $('#tudo').fadeOut(1500);
    }
});
...
```

[arquivo-8.6a.html]

8.7 Animação personalizada com `animate()`

Os métodos de animação estudados até aqui, neste capítulo, são nativos da biblioteca jQuery. Contudo, você não está limitado ao uso somente dessas animações, pois a biblioteca fornece um método de animação genérico para ser utilizado em animações personalizadas. Trata-se do método `animate()`, que permite manipular propriedades CSS numéricas. Para demonstrar essa animação, use os códigos mostrados a seguir:

▶ **CSS**:

```
1.    <style type="text/css" media="all">
2.        body {/* sem alterações */}
3.        label {/* sem alterações */}
4.        p {background:#444;color:#fff; padding:8px 20px;}
5.        button {/* sem alterações */}
6.    </style>
```

Código comentado:

Linha(s)	Descrição
Linhas 1 a 6	Altere sua folha de estilo em função da retirada de elementos da marcação. Use um só parágrafo com fundo cinza-escuro e letras brancas conforme a regra CSS na linha 4.

▶ **jQuery**:

```
1.    <script type="text/javascript" src="../jquery-1.2.6.js"></script>
2.    script type="text/javascript">
3.       $(document).ready(function() {
4.          $('html').css('height', '101%')
5.          $('h1').after('<button type="button">Esconder</button>');
6.          var emAndamento = '<img src="carregando.gif" class="gif-andamento" />';
7.          var alturaJanela = $(window).height();
8.          var alturaObjeto = $('p').outerHeight();
9.          var posicaoObjeto = $('p').position().top;
10.         var amplitudeDeslocamento = alturaJanela - alturaObjeto - posicaoObjeto;
11.      $('button').click(function() {
12.         if ($(this).text() == 'Mostrar') {
13.            $(this).text('Esconder').hide().after(emAndamento);
14.            $('p').show().animate({top:0, opacity:1}, 2500, function() {
15.                  $('.gif-andamento').css('display', 'none');
16.                  $('button').show();
17.            });
18.         } else {
19.            $(this).text('Mostrar').hide().after(emAndamento);
20.            $('p').css({position:'relative'})
21.               .animate({top:amplitudeDeslocamento, opacity:0}, 2500, function() {
22.                  $('p').hide();
23.                  $('.gif-andamento').css('display', 'none');
24.                  $('button').show();
25.               });
26.            };
27.         });
28.      });
29.   </script>
```

[arquivo-8.7a.html]

Para exemplificar a animação personalizada, utilizou-se um elemento parágrafo que será deslocado para fora da tela com um efeito do tipo cair. Para restabelecer a visibilidade, o efeito será ao contrário, ou seja, o parágrafo entrará na tela com um efeito do tipo subir. Como exercício, você pode incrementar essa animação adicionando o efeito de transição de opacidade. Outro exercício interessante consiste em deslocar o objeto para fora da tela, fazendo-o sair pela lateral. E que tal sair pela direita e entrar pela esquerda?

Código comentado:

Linha(s)	Descrição
Linha 4	Esta regra CSS força o aparecimento da barra de rolagem vertical nos navegadores-padrão, mesmo sem necessidade dela, ao contrário do Internet Explorer que reserva o espaço independentemente do espaço vertical requerido pelo conteúdo. Nessa animação, o objeto é colocado para fora da tela por sua parte inferior e, nesse momento, aparece a barra de rolagem, o que causa um deslocamento horizontal, como um pulo lateral, da página. Forçando o aparecimento da barra de rolagem, corrige-se esse pulo. Experimente retirar essa linha do código e visualize a animação para ver o desagradável pulo.
Linha 5	Inseriu-se um botão logo após o elemento `h1`, que, ao ser clicado, irá executar a animação.
Linha 6	Armazenou-se, na variável `emAndamento`, uma marcação HTML que define a inserção de uma gif-animada indicativa de carregamento. Para uma ferramenta on-line geradora desses tipos de gif, visite: `http://www.ajaxload.info/`.
Linha 7	Armazenou-se, na variável `alturaJanela`, a altura total da janela do navegador em unidade pixel.
Linha 8	Armazenou-se, na variável `alturaObjeto`, a altura total, em unidade pixel, do box CSS que contém o objeto a animar, neste caso, o parágrafo. Veja [C4 S4.3].
Linha 9	Armazenou-se, na variável `posiçãoObjeto`, a coordenada vertical, ou seja, a distância em pixel que o objeto se encontra do topo da janela do navegador.
Linha 10	Armazenou-se, na variável `amplitudeDeslocamento`, a distância em pixel que o objeto deverá percorrer desde sua posição atual até sair completamente da tela por seu limite inferior (Figura 8.11).
Linha 11	Condicional `if() {}` testa a situação da página e decide o que fazer. Na situação inicial, o teste resulta falso, pois o texto do botão é Esconder e não Mostrar. Assim a execução do script vai para a linha 19.
Linha 19	Alterou-se o texto do botão para Mostrar e escondeu-se o botão. Ao deixar o botão à mostra, possibilita ao usuário clicá-lo durante a animação, o que irá causar confusão. Altere o script e comprove você mesmo. A seguir, insere-se, no lugar do botão, a gif-animada indicando ao usuário que a animação está em andamento.
Linha 20	Atribui-se `position:relative` ao objeto a animar, pois como se sabe das técnicas CSS, somente objetos posicionados são passíveis de ser deslocados por definição de coordenadas.

Linha(s)	Descrição (cont.)
Linhas 21 a 25	A animação começa aqui, com a passagem de parâmetros para o método `animate()`. Nessas linhas, a ação é a seguinte: anime o parágrafo cuja posição se acabou de definir como relativa, levando-o para uma coordenada vertical igual à quantidade de pixels necessária para fazê-lo sair da tela pela parte inferior (variável `amplitudeDeslocamento`); a animação deverá durar dois segundos e meio (2500). Ao terminar a animação, retire o parágrafo da marcação (linha 33), mostre o botão agora com texto "Mostrar" (linha 22) e esconda a gif-animada (linha 23), indicando o fim da animação. Agora o botão está com o texto "Mostrar" e a condicional, na linha 12, resulta verdadeira, rodando o script a partir da linha 13.
Linha 13	O funcionamento segue a mesma lógica da explicada anteriormente, para a condição de teste falsa, e agora o script coloca o objeto à vista.

É necessário esconder o parágrafo conforme mostra a linha 22, pois o deixando à vista, após ter saído da tela, uma simples rolagem vertical da tela o revelaria ali, o que certamente não se deseja, mesmo porque havendo conteúdo após a borda inferior da tela, o parágrafo se colocaria sobre tal conteúdo.

Na figura 8.11, há um esquema indicativo do cálculo do deslocamento vertical necessário para fazer um objeto sair pela parte inferior da tela.

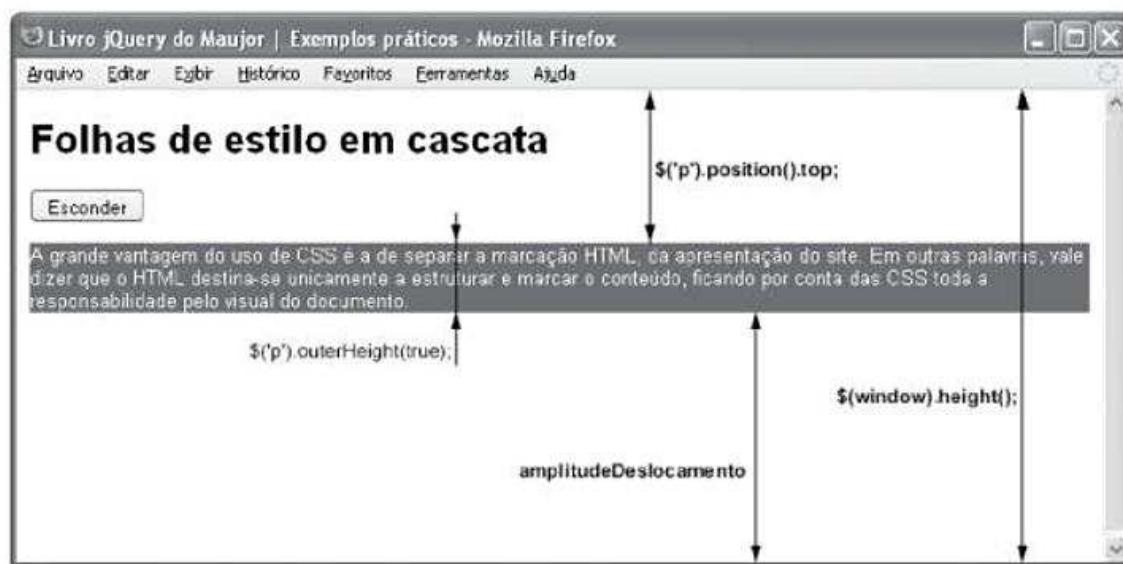



Figura 8.11 – Cálculo do deslocamento vertical.

Consulte o `arquivo-8.7b.html` para visualizar os cálculos dos valores para as variáveis mostradas nesse exemplo. Abra o arquivo e proceda a alguns redimensionamentos da janela do navegador, recarregando a página a cada redimensionamento, para visualizar os resultados dos cálculos.

CAPÍTULO 9
Revelando conteúdos

Neste capítulo, serão estudados alguns efeitos da biblioteca jQuery destinados a esconder e revelar conteúdos. Não se pode esquecer que efeitos obtidos com scripts rodando no lado do cliente, tais como são os efeitos jQuery, devem ser projetados de modo a não bloquear nem obstruir o acesso a conteúdos. É o que se denomina JavaScript não obstrutivo.

Um bom efeito sempre começa a partir de um documento bem estruturado, com sua marcação e CSS validadas no W3C e acessível. Ao longo do processo de criação de um efeito, todo cuidado é necessário para não introduzir regras CSS, ou marcação, que venham a comprometer a acessibilidade.

Esconder e revelar conteúdos é um efeito muito fácil de obter, podendo ser empregado com segurança nas situações em que extensos conteúdos obrigam o usuário a rolagens de tela. Esse caso é um clássico exemplo do que se denomina "enriquecimento da experiência do usuário", caracterizando-se por proporcionar ao usuário com infra-estrutura apropriada (por exemplo: JavaScript habilitado no navegador) melhores condições de uso do documento em relação aos usuários sem a necessária infra-estrutura. E, aqui, vale repetir: o acesso é garantido a todos, mas os efeitos de enriquecimento irão se destinar apenas aos habilitados a experimentá-los.

9.1 FAQ CSS

Para as explicações dos efeitos de esconder e revelar conteúdos, será tomada como base uma adaptação de trecho de uma FAQ CSS real publicado em `http://www.maujor.com/tutorial/faq.php`.

Escolheu-se uma FAQ CSS, mas evidentemente as técnicas mostradas não são de aplicação exclusiva para um documento que apresente uma listagem de perguntas e respostas como são as FAQs. Ao contrário, entendendo a técnica explicada para esse exemplo, você facilmente poderá adaptá-la a qualquer tipo de conteúdo.

Marcação básica

Para desenvolver o efeito revelar e esconder da FAQ adotada como exemplo, será utilizada uma marcação HTML básica na qual se escolhe um total de dez perguntas. Veja a seguir essa marcação devidamente comentada.

Para melhor entendimento das técnicas de desenvolvimento, optou-se por iniciar com um script simples em uma primeira etapa e, a partir daí, ir modificando, aperfeiçoando e apresentando alternativas ao script em etapas sucessivas.

O foco principal deste capítulo é o aprendizado das técnicas de desenvolvimento com jQuery. Vários são os caminhos para se conseguir um mesmo efeito. Certamente você, à medida que progredir no estudo, encontrará alternativas para os scripts apresentados, não devendo limitar-se a um simples copiar e colar. Agindo assim, estará enriquecendo e aperfeiçoando seu aprendizado.

A marcação HTML e CSS da página básica para os exemplos é mostrada a seguir.

▶ **CSS**:

```
1.    <style type="text/css" media="all">
2.    body {
3.        width:400px;
4.        font:80%/1.2 Arial, Helvetica, sans-serif;
5.        margin:0 auto;
6.        padding:0;
7.    }
8.    dt {
9.        margin:20px 0 3px 0;
10.       font-weight:bold;
11.       border-top:1px solid #ccc;
12.   }
13.   dd {
14.       margin:0;
15.       padding:5px 0 0 0;
16.       text-align:justify;
17.   }
18.   </style>
```

Trata-se de uma folha de estilos mínima destinada a centralizar a página e inserir apresentação básica com a finalidade exclusiva de facilitar a visualização do exemplo.

▶ **HTML**:

```
1.    <h1>FAQ CSS</h1>
2.       <h2 id="indice">Índice</h2>
3.       <ol>
4.          <li><a href="#css">O que significa a sigla CSS?</a></li>
5.          <!-- omitidos oito itens -->
6.          <li><a href="#comentario">Posso incluir comentários nas regras CSS?</a></li>
7.       </ol>
8.       <h3>Perguntas e respostas</h3>
9.       <dl>
10.         <dt id="css">O que significa a sigla CSS?</dt>
11.         <dd>CSS é a sigla...</dd>
12.         <dd><a href="#indice">Índice</a></dd>
13.         <!-- omitidos oito itens -->
14.         <dt id="comentario">Posso incluir comentários nas regras CSS?</dt>
15.         <dd>Sim. Comentários podem e ...></dd>
16.         <dd><a href="#indice">Índice</a></dd>
17.      </dl>
18.   </body>
19.   </html>
```

[arquivo-9.1a.html]

Código comentado:

Linha(s)	Descrição
Linha 1	Título da página.
Linha 2	Título para o índice das perguntas contidas no FAQ. Note a presença do atributo `id="indice"` nesse título. Esse `id` servirá como âncora de destino para a navegação interna que remete o usuário ao índice sempre que se clicar qualquer dos links denominados "índice" existentes ao final de cada resposta. Deve-se evitar a prática de marcar destinos de links internos com a sintaxe HTML `<a name="nome"></a>`, a não ser nos casos de documentos servidos para navegadores antigos que não suportam `id` para marcar âncoras. Assim, prefira o uso de `id` para marcar âncoras de destino da navegação interna.
Linhas 3 a 7	Marcação de uma lista ordenada cujos itens contêm as perguntas da FAQ.

Linha(s)	Descrição (cont.)
Linhas 9 a 17	Marcação de uma lista de definição contendo as respostas para cada uma das perguntas. Listas de definição são aquelas marcadas com o elemento HTML `dl` (definition list). Os dois elementos desse tipo de lista são: `dt` (definition term), que significa termo a ser definido, e `dd` (definition description), que significa descrição da definição para o termo. Trata-se de uma lista na qual se define um termo e, a seguir, descreve(m)-se uma ou mais características do termo. Esse tipo de lista foi escolhido para marcar a FAQ adotada como exemplo, pois se considera cada pergunta como o termo a ser definido e a resposta, a definição ou descrição para a pergunta. Ao final de cada resposta, cria-se um link para o índice, marcado com um termo de definição. Para não alongar o código, mostra-se somente a marcação para duas das dez perguntas existentes no exemplo.

A renderização da página básica para o FAQ é apresentada na figura 9.1.

Figura 9.1 – Página básica para FAQ.

Tal documento encontra-se em sua forma mais elementar com alguma estilização básica, contudo um usuário ainda que desabilite suporte às CSS em seu navegador ou abra a página em um navegador de texto, terá acesso a todas as perguntas e respostas do FAQ e nenhum conteúdo será perdido.

esses assuntos aos quais se dispensa a máxima atenção. Assim, no caso da FAQ em questão, é de vital importância que o usuário identifique instantaneamente como revelar o conteúdo.

Na solução dada, os links para o índice continuarão funcionais. Tais links serão particularmente úteis nos casos em que o usuário tiver de rolar a página para visualizar respostas localizadas abaixo da dobra (linha limite inferior da janela do navegador).

Cada vez que o usuário clicar um dos links para ver a resposta, esta será revelada. Adicionalmente, acrescentam-se uma borda e uma cor de fundo para destacar a resposta revelada.

Observe, na figura 9.3, o que aconteceu após o usuário ter clicado sucessivamente os links para revelar as respostas a duas perguntas. Note que acima da resposta se disponibilizou um link para fechá-la e abaixo dela, um link que remete ao índice.

Veja, nas figuras 9.2 e 9.3, o ciclo completo do efeito que se pretende obter nessa primeira etapa.

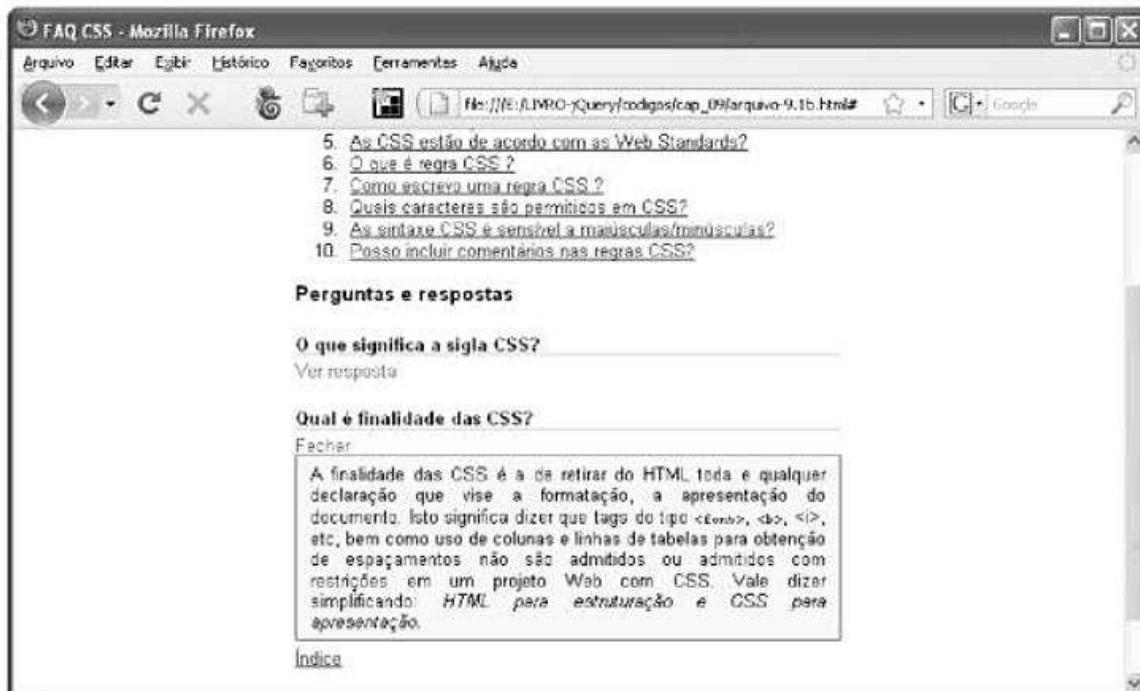



Figura 9.3 – Página para FAQ: primeira etapa – revelar.

Veja a seguir os códigos para desenvolver o efeito proposto para esta etapa.

Não se irão acrescentar nenhuma marcação nem regra CSS ao documento. Toda a marcação necessária e a folha de estilo serão inseridas via script.

Será que esta é uma boa escolha? Afinal se estão misturando marcação e apresentação com comportamento. É boa prática manter essas três camadas do desenvolvimento independentes. Toda a marcação fica no HTML da página, a apresentação, em arquivos de folhas de estilo e o comportamento, em arquivos de script.

Quando se trata de usar jQuery e scripts de maneira geral, com a finalidade de incrementar a apresentação da página, nada há de errado em se adicionar marcação e estilização via script. Uma criteriosa escolha do que deve ser inserido e, principalmente, o uso de bom senso pelo desenvolvedor são mais importantes que despender tempo com discussões dessa ordem. O objetivo principal do uso de script é enriquecer a experiência do usuário, sem bloquear seu acesso a conteúdos. Para cumprir esse objetivo, é perfeitamente válido inserir marcação e estilização via script.

Um erro comum e frequente relacionado a esse princípio é esconder, inicialmente, o conteúdo com uso de CSS e revelá-lo via script. O procedimento deve ser exatamente o inverso, ou seja, deixá-lo visível e definir regra CSS para esconder via script. Agindo assim, não se bloqueia o acesso ao conteúdo a usuários que estejam navegando sem JavaScript habilitado.

Nessa etapa, será utilizado o script para tais inserções e, na sequência do desenvolvimento, em etapas futuras, serão mostradas inserções em sua respectiva camada, misturadas com inserções por script, todas sendo soluções válidas e funcionais.

Em todos os exemplos desenvolvidos neste livro, colocaram-se as folhas de estilo e os scripts incorporados na seção head do HTML da página unicamente para facilitar ao leitor o estudo dos códigos-fonte. Em desenvolvimento real, deve-se linkar tanto folhas de estilo como scripts.

Observe o script que produz o efeito proposto e os comentários:

```
1.     <script type="text/javascript">
2.     $(document).ready(function() {
3.         var textoVer = '<a href="#" style="color:#666; text-decoration:none;">
               Ver resposta</a>';
4.         $('dd').css('display', 'none');
5.         $('dt').after(textoVer);
6.         $('a').click(function() {
7.             if ($(this).text() == 'Ver resposta') {
8.                 $(this).next().next().toggle();
9.                 $(this).text('Fechar');
10.                $(this).next()
11.                .css({
```

```
12.                 border:'1px solid #c30',
13.                 padding:'5px 10px',
14.                 background:'#ff9'
15.             })
16.             .slideToggle('slow');
17.         } else {
18.             $(this). text('Ver resposta');
19.             $(this).next().slideToggle('slow');
20.             $(this).next().next().toggle();
21.         }
22.     });
23. });
24. </script>
```

 [arquivo-9.1b.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Criou-se uma variável para conter a marcação HTML para os links que irão revelar e esconder as respostas.
Linha 4	O script determina o seguinte: encontre todos os elementos `dd` no documento e, para estes, vale a declaração CSS definindo que não sejam renderizados, ou seja, esconda-os. Esta linha faz que todas as respostas e os links para o índice desapareçam da página. Faça uma cópia do arquivo, retire todo o script da página, exceto esta linha, mas, cuidado, não retire as linhas 1, 2, 23 e 24 que são o container do script. Salve o arquivo com o nome `arquivo-9.2b.teste.html` e visualize-o no navegador.
Linha 5	Aqui a ação é a seguinte: encontre todos os elementos `dt` (note que os elementos `dt` marcam as perguntas) da página e insira logo após cada um deles a marcação HTML para o link que irá mostrar as respostas (definida na linha 3). Agora é como se você tivesse esse link imediatamente depois de cada uma das perguntas. Tais elementos não estarão escondidos por enquanto. Abra o arquivo `arquivo-9.2b.teste.html` que salvou conforme sugerido no item anterior e acrescente nele as linhas 3 e 5 do script. Salve-as e visualize no navegador. Agora, além das perguntas, aparece logo abaixo delas o link "Ver resposta". Clique o link e nada acontecerá.
Linha 6	Lembre-se de que o elemento `a` acaba de ser inserido na marcação, logo após o elemento `dt`, e nesta linha está dito: quando for clicado o elemento `a`, isto é, "Ver resposta", execute a função que se segue.

Linha(s)	Descrição (cont.)
Linha 7	Esta linha declara que se efetue um teste condicional e, dependendo do resultado, o script executará uma de duas ações possíveis. O teste é o seguinte: se o conteúdo textual do elemento `a` for "Ver resposta", execute o que se segue, senão vá para a linha 17 (declaração `else`). Aqui o seletor `$(this)` tem o mesmo significado que em JavaScript, ou seja, neste caso se refere ao elemento que acaba de ser clicado (`a`). O método `text()` recupera o conteúdo textual de um elemento. O teste resulta verdadeiro, pois inicialmente o texto do elemento `a` é "Ver resposta", e assim o script em questão continua na linha seguinte.
Linha 8	Aqui está dito o seguinte: encontre o elemento que vem depois do elemento `a` que você clicou e, a seguir, o elemento que vem depois. Esclarecendo: trata-se do segundo elemento após o `a`. Sabendo que o elemento `a` está logo após o elemento `dt` e examinando a marcação, verifica-se que depois do elemento `a` vem um elemento `dd` com a resposta e, depois desse outro elemento, `dd` com o link para o índice. É esse link o alvo desse encadeamento e sobre ele aplicar-se-á o método `togle()`, ou seja, mude a visibilidade dele. Como se inicia escondido (veja a linha 4), aqui será revelado. Resumindo: se o usuário clicar "Ver resposta", revele o link para o índice.
Linha 9	Aqui se está substituindo o conteúdo do elemento `span`, que atualmente é "Ver resposta", por "Fechar", ou seja, ao se revelar a resposta, o link passará de "Ver resposta" para "Fechar".
Linhas 10 a 15	Definição das declarações de estilos para os conteúdos a ser revelados. Aqui está dito o seguinte: aplique as regras de estilo ao elemento que se segue ao elemento clicado. O elemento que contém o link clicado é um `span` e, logo depois dele, encontra-se o elemento `dd` que contém a resposta à pergunta. As regras de estilo definem uma borda, um `padding` e uma cor de fundo para o texto da resposta a ser revelada.
Linha 16	Nesta linha, definiu-se o efeito de revelação do conteúdo. Optou-se por mostrar o conteúdo fazendo que apareça segundo um movimento vertical de cima para baixo. Salve uma cópia do `arquivo-9.1b.html`, substitua essa linha por `fadeIn(2000)` e veja o resultado no navegador, depois substitua por `show(1000)` e veja o resultado. Nessa linha termina a execução do script determinado pelo resultado do teste condicional iniciado na linha 7 e o script pula os comandos dentro de `else` na linha 17, não encontrando nada mais depois dele, encerrando-se a ação desencadeada pelo script. Assim, a situação atual é a seguinte: a resposta esta à vista, anterior a esta existem um link "Fechar" e, depois dela, um link "Índice". Se o usuário clicar o link "Índice", simplesmente irá para a respectiva pergunta no início da página. Se clicar o link "Fechar", estará ativando novamente o script na sua linha 6. Agora o ciclo se repete com a diferença de que o teste condicional resulta falso.

Linha(s)	Descrição (cont.)
Linha 6	O clique do usuário agora ocorreu no elemento a contendo o texto "Fechar" (veja o comentário anterior para a linha 9).
Linha 7	Agora o teste resultou falso, pois o texto do link não é "Ver resposta", e sim "Fechar". Em consequência, o script vai para a linha 17, que marca o início das ações para o caso de a condicional retornar falsa.
Linha 18	Clicado o link "Fechar", a primeira ação é a troca do texto do link de "Fechar" para "Ver resposta".
Linha 19	Esta linha estabelece o seguinte: esconda a resposta usando o efeito de desaparecer segundo um movimento vertical de baixo para cima. Aqui se sugere que você modifique o script, na cópia feita anteriormente, usando `fadeOut()` e `hide()` e, depois, usando um tipo de efeito para revelar e outro para mostrar.
Linha 20	Aqui se está escondendo o link para o índice. Experimente retirar esta linha e ver o resultado no navegador.

Terminado o desenvolvimento e com a página funcionando como planejado, faça o seguinte: abra a página no navegador, desabilite JavaScript e recarregue a página. Algum conteúdo se perdeu? Algum conteúdo estranho à página apareceu?

Lembre-se de que a página deve permanecer funcional e lógica mesmo com JavaScript desabilitado. Nada de desabilitar JavaScript e aparecer na página um link ou um texto do tipo "Ver Resposta" ou "Fechar". Muito menos desaparecer uma resposta, ou todas, ou o índice. Isto é JavaScript não obstrutivo usado para incrementar a experiência do usuário.

Pode-se, então, concluir que ao final desta primeira etapa o efeito que se acabou de desenvolver trouxe como resultado uma página bem mais interessante e com apelo visual bem melhor que o da página básica. Contudo, alguns pontos podem e devem ser melhorados.

Segunda etapa

Nesta etapa, o objetivo será retirar a estilização e marcação do script em questão. Um dos fundamentos do desenvolvimento, segundo os Padrões Web, preconiza independência entre camadas de marcação, apresentação e comportamento.

A propósito da afirmativa anterior, é interessante fazer um comentário.

Uma primeira corrente de desenvolvedores preconiza e defende veementemente uma interpretação dos Padrões Web de forma absolutamente rígida e

sem abrir concessões. Não admitem, em hipótese alguma o uso de elementos adicionais na marcação, defendem o minimalismo do código e são contrários a qualquer mistura das camadas de apresentação, marcação e comportamento, enfim, são rígidos na interpretação da letra das normas. A segunda corrente de desenvolvedores considera os Padrões Web uma bobagem e, ainda, desenvolve sites com tabelas e scripts ultrapassados. A terceira corrente tem consciência dos Padrões Web, conhece seus fundamentos e os aplica em seu dia a dia de trabalho, contudo não abrindo mão de documentos válidos se permitem concessões, como o uso de um `div` a mais no código para encurtar o caminho para a consecução de determinado objetivo. Você é livre para aderir a qualquer um dos três grupos, mas os códigos desenvolvidos neste livro seguem a filosofia dos grupos não extremistas, que parece a mais apropriada ao mundo real do desenvolvimento. Rigidez de interpretação é uma postura teórica e utópica. Ignorar os Padrões Web é se declarar ultrapassado.

Inicie identificando, no script anterior, quais as linhas que podem ser movidas para outra camada. São elas:

```
3.      var textoVer = '<span><a href="#" style="color:#666;
            text-decoration:none;">Ver resposta</a></span>';
4.          $('dd').css('display', 'none');
10.         $(this).next()
11.             .css({
12.                 border:'1px solid #c30',
13.                 padding:'5px 10px',
14.                 background:'#ff9'
```

A linha 3 contém marcação e estilização. Passe a marcação para o HTML da página e a estilização para as CSS. A marcação para os links deve ser inserida logo após cada elemento `dt`. Veja a seguir as alterações a introduzir.

▶ **HTML**:

```
...
<h3>Perguntas e respostas</h3>
<dl>
    <dt id="css">O que significa a sigla CSS?</dt>
        <dd>Ver resposta</dd>
        <dd>Fechar</dd>
        <dd>CSS é a sigla para Cascading Style ...
...
```

Fazendo isso, veja, na figura 9.4, o que acontece com a página básica.

Figura 9.4 – Página para FAQ: segunda etapa – marcação.

Acabou-se de cometer, propositadamente, um engano muito comum quando se desenvolve sem pensar Padrões Web. Introduziram-se componentes necessários ao funcionamento do script em questão, na renderização da página. E o que acontece quando JavaScript está desabilitado? O que fazem aqueles links na página? Absolutamente nada! Retire-os dali rapidamente.

Embora esta não seja a solução, veja comentários adicionais sobre essa inserção de código. Talvez você esteja se perguntando o porquê de não se ter sugerido o uso de um elemento neutro como `span` ou, mais apropriadamente, o elemento `a`, que é o semanticamente correto para marcar o `link`.

Se fizer isso, estará invalidando sua marcação HTML, pois o elemento `dl` admite somente os elementos `dt` e `dd` como elementos-filho. Não se pode acrescentar nenhum outro elemento dentro dele. Na sequência do exemplo, você irá efetivamente usar o elemento `dd` para container do link em uma solução um pouco diferente da sugerida aqui e poderá, então, constatar por si mesmo o que acontece, além de um código inválido, se usar outro elemento que não `dd`. Adiante, esse assunto será retomado.

O problema é o seguinte: você precisa do elemento `dd` naquela posição, mas não pode permitir que o texto seja renderizado. Você seria capaz de apontar a solução?

navegador, neste caso, não constrói o DOM corretamente em consequência da marcação inválida.

Em qualquer situação de desenvolvimento de documentos para a web, valide constantemente seu código ao longo do processo de criação. Sempre que tiver o controle da manutenção de um documento já existente, valide-o também.

Sugestão: salve uma cópia do `arquivo-9.1c.html`, modifique a marcação substituindo `dd` por `span` ou `a` e visualize o resultado no Firefox e no Internet Explorer.

Antes de passar para a etapa seguinte, desabilite JavaScript no navegador e visualize sua criação. Tudo certo?

Se o usuário estiver navegando com o uso do teclado, a página será inacessível a ele, a menos que desabilite JavaScript. Experimente navegar com o uso do teclado na página gerada pelo arquivo da solução apresentada na primeira etapa e, depois, vá à página do arquivo dessa etapa e realize o mesmo procedimento. Você vai constatar que, na primeira, você abre as respostas com o teclado normalmente e, na segunda, não. Elementos que não são links não são acessíveis via teclado, a menos que você crie script para tal. Como exercício, modifique o script dessa solução para torná-la acessível via teclado.

Terceira etapa

Nesta etapa, será efetuada uma mudança na apresentação dos links para abrir e fechar as respostas. Em vez de links textuais, será utilizada uma imagem como link. É comum encontrar, na web, imagens de um sinal de mais e menos (+/-) para tais links. Na solução que será desenvolvida, você utilizará a imagem de olho aberto e fechado. Não está em questão o design da solução, o que importa é o objetivo final do estudo: mostrar a técnica de uso de uma imagem. Conhecendo a técnica, você pode usar a imagem que melhor se adapte a seu projeto.

Complementarmente, nesta solução, ao contrário da anterior, considere a navegação via teclado. Para que um elemento seja acessado via teclado, há duas opções de marcação: marcar com o elemento `a` (elemento para marcar links) ou definir o atributo `tabindex` que se destina a estabelecer uma ordem de tabulação.

O uso do atributo `tabindex` é válido para controles de formulário (por exemplo: `input`, `textarea` e `button`). Se você definir esse atributo para um elemento parágrafo, por exemplo, este será acessado pelo teclado normalmente, contudo sua marcação não será validada. Priorize a validação e, assim, opte por usar um link em elementos que devam ser acessados pelo teclado.

No exemplo em questão e na maioria dos scripts que dependem de um clique do ponteiro do mouse para disparar uma ação, usar um link é, sem dúvida, uma solução semanticamente correta. A finalidade de marcar como link é proporcionar acesso via teclado, então se criará um link morto. O destino de um link morto é o sinal tralha ou jogo-da-velha (#) e a marcação segue a sintaxe mostrada a seguir:

```
<a href="#">Link morto</a>
```

Com essas informações e tendo entendido o funcionamento do script desenvolvido na etapa anterior, crie um script para revelar e esconder as respostas da FAQ com um clique em uma imagem.

Veja, na figura 9.5, o resultado final proposto para essa etapa.

Figura 9.5 – Página para FAQ: terceira etapa.

As regras CSS necessárias a esta solução são as seguintes:

```
.estilo-um {/* Já usada na solução anterior, estiliza o texto da resposta revelada */
       border:1px solid #c30;
       padding:5px 10px;
       background:#ff9;
       }
img {border:none; display:none;}
```

A novidade aqui é o uso de uma regra CSS para retirar a borda azul padrão que os navegadores colocam em volta de uma imagem quando esta é um link. Lembre-se de que você irá utilizar a imagem como se fosse um link morto para proporcionar acesso via teclado. Faça uma cópia do arquivo desta etapa, retire essa regra CSS, salve o arquivo e visualize-o no navegador. Constate o aparecimento de uma borda horrível em volta de cada um dos (não tão lindos e azuis) olhos.

A marcação para cada resposta será como a mostrada a seguir:

```
<h3>Perguntas e respostas</h3>
<dl>
    <dt id="css">O que significa a sigla CSS?</dt>
        <dd><a href="#"><img src="" alt="" /></a></dd>
        <dd>CSS é a sigla para ...</dd>
        <dd><a href="#indice">Índice</a></dd>
...
```

Tal como se fez para o link que leva ao índice, inseriu-se marcação para servir de container à imagem do olho. Note que o atributo `src` da imagem está vazio. Por que não optar por simplesmente omitir o atributo? Porque a página não passaria no validador, pois esse atributo é obrigatório. Mas a inserção dele, ainda que vazio, irá acarretar um problema que será tratado a seguir.

Você seria capaz de dizer qual a finalidade da declaração `display:none`, mostrada anteriormente na regra CSS, para a imagem? Por que esconder a imagem se precisará dela na página?

A razão é a seguinte: quando há uma imagem inserida na marcação cujo atributo `src` leva a um endereço no qual não existe uma imagem, o navegador Internet Explorer renderiza um ícone padrão, para indicar a existência de uma imagem quebrada. Se JavaScript for desabilitado no navegador, tal ícone irá aparecer no Internet Explorer.

O script jQuery é mostrado a seguir:

```
1.      <script type="text/javascript">
2.          $(document).ready(function() {
3.              var olhoAberto = 'olho_aberto.gif';
4.              var olhoFechado = 'olho_fechado.gif';
5.              var olho = $('dd > a > img');
6.              $('dd + dd').hide();
7.              $(olho).attr('src', olhoFechado).show();
8.                  $(olho).click(function(event) {
9.                      event.preventDefault();
10.                     if ($(this).attr('src') == olhoFechado) {
```

```
11.                  $(this).attr('src', olhoAberto);
12.                  $(this).parent().parent().next().next().show();
13.                  $(this).parent().parent().next().addClass('estilo-um')
14.                  .slideToggle('slow');
15.              } else {
16.                  $(this).attr('src', olhoFechado);
17.                  $(this).parent().parent().next().next().hide();
18.                  $(this).parent().parent().next().addClass('estilo-um')
19                   .slideToggle('slow');
20.              }
21.          });
22.      });
23.  </script>
```

 [arquivo-9.1d.html]

Código comentado:

Linha(s)	Descrição
Linhas 3 e 4	Criaram-se duas variáveis para armazenar o caminho para a imagem do olho, com a finalidade de economizar digitação posterior no script. Note que, neste caso, a imagem foi gravada na mesma pasta que o arquivo e talvez você esteja considerando desnecessária a criação das variáveis. Imagine esta situação que vem se tornando muito popular na construção de sites: contratar serviços de terceiros para hospedar imagens. Nesses casos, o caminho para a imagem, normalmente, é um endereço enorme repleto de números e as variáveis criadas se justificam.
Linha 5	Criou-se uma variável para definir um seletor jQuery que tem como alvo a imagem do olho. Note, na marcação mostrada anteriormente, que o elemento-alvo imagem é filho do elemento `a`, que, por sua vez, é filho do elemento `dd`, daí o seletor `dd > a > img`.
Linha 6	Aqui está dito o seguinte: esconda todo elemento `dd` que se encontra imediatamente após um elemento `dd`. Em cada resposta, há três elementos `dd` que se seguem, então é fácil concluir que este método jQuery irá esconder o segundo (contém as respostas) e o terceiro (contém o link para o índice) elemento `dd`, deixando à vista o primeiro (contém a imagem do olho).
Linha 7	Nesta linha, jQuery seleciona todos os elementos da variável olho (os elementos imagens inseridos na marcação) e neles coloca o atributo `src` com o valor da imagem de um olho fechado. A seguir, torna esse elemento visível, pois estava oculto por regra CSS em razão da renderização do ícone indicador de imagem quebrada no IE como explicado anteriormente. O script é executado até aqui quando a página é carregada e aguarda um clique no olho para continuar. O script fez o seguinte: escondeu a resposta e o link para o índice (linha 6) e inseriu o olho fechado (linha 7).

Quarta etapa

Seria muito bom se a FAQ em questão oferecesse a facilidade ao usuário de ele não ter que clicar o link fechar toda vez que passasse para a visualização de outra resposta. Na verdade, ele não é obrigado a fechar uma resposta para revelar outra, podendo abrir as respostas uma de cada vez, mas isso aumentará o conteúdo na tela e a rolagem vertical também, enchendo a tela de respostas e dificultando a consulta a uma resposta revelada anteriormente.

Nesta etapa, é fundamental que o usuário aperfeiçoe seu script de modo a sempre que uma resposta for revelada, a anterior que tenha sido revelada fechará automaticamente. Desta forma, haverá sempre uma resposta na tela a cada clique do usuário.

Suponha que o usuário tenha iniciado a leitura com a primeira resposta e, a seguir, clicado para abrir a segunda resposta. Veja, na parte superior da figura 9.6, o instante logo após o clique para abrir a segunda resposta e, na tela inferior, o fim da animação prevista no script.

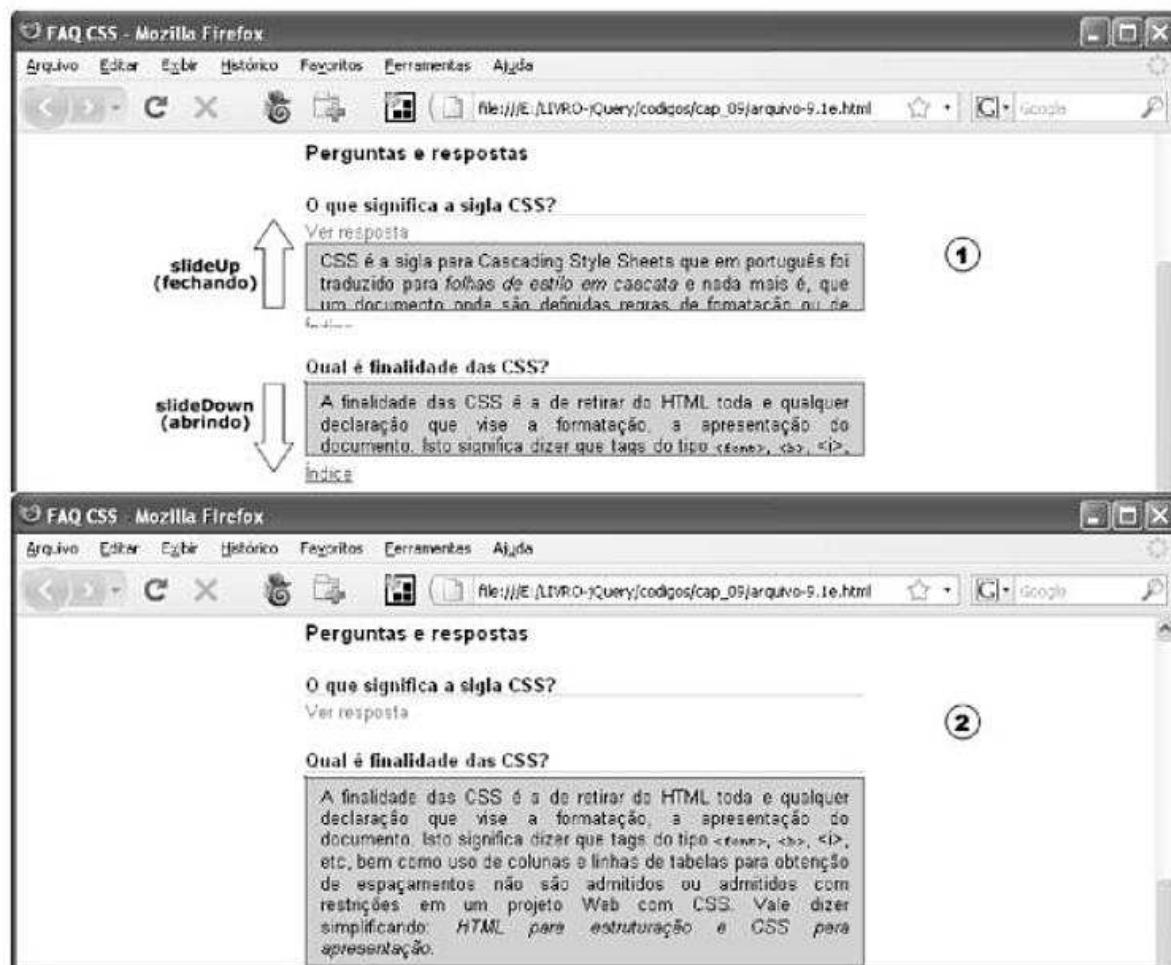



Figura 9.6 – Página para FAQ: quarta etapa.

A regra CSS utilizada para esta solução é a mesma que se usou para a classe `estilo-um`, que define borda, `padding` e cor de fundo para a resposta.

O script jQuery é mostrado a seguir:

```
1.    <script type="text/javascript">
2.        $(document).ready(function() {
3.            var textoVer = '<a href="#" style="color:#666; text-decoration:none;"
                  class="ver">Ver resposta</a>';
4.            $('dd').css('display', 'none');
5.            $('dt').after(textoVer);
6.            $('.ver').click(function(event) {
7.                event.preventDefault();
8.                $(this).text('');
9.                $(this).next()
10.               .addClass('estilo-um')
11.               .slideDown(1500)
12.               .siblings('dd:visible').slideUp('fast');
13.               $(this).next().siblings('dd:visible').prev().text('Ver resposta');
14.           });
15        });
16.   </script>
```

[arquivo-9.1e.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Criou-se uma variável para armazenar o link para ver a resposta. Aqui, optou-se por criar a marcação e a estilização via jQuery e atribuiu-se à classe denominada `ver` o elemento link. Utilizou-se o elemento `a` para links, com a finalidade de garantir acesso via teclado. Note que, nesta solução, não foi necessário um link para fechar a resposta, já que uma resposta revelada fechará, de modo automático, outra eventualmente aberta.
Linha 4	Esta você conhece das etapas anteriores.
Linha 5	Inseriu-se a marcação HTML do link para ver a resposta, imediatamente após o elemento `dt` que contém a pergunta.
Linha 6	Ao se clicar o elemento com a classe denominada `ver` (o link para ver a resposta), o script entra em ação.
Linha 7	Tem o mesmo efeito de `return false`; da linguagem JavaScript, evitando que o navegador siga o link; no caso em questão, a janela rola para o topo.
Linha 8	Retira o texto do link para mostrar a resposta.

Linha(s)	Descrição (cont.)
Linha 9	Seleciona o elemento que se segue na marcação ao elemento clicado. É o elemento `dd` que contém a resposta. Não esqueça que elementos inseridos com uso do script passam a fazer parte do DOM, ou seja, tudo se passa como se efetivamente estivessem presentes na marcação, embora você não consiga visualizá-los no código HTML da página.
Linha 10	Estiliza, adicionando uma classe que consta das CSS, o texto da resposta.
Linha 11	Revela a resposta.
Linha 12	Seleciona todos os elementos-irmão do elemento `dd` – `$this).next()` – que sejam também `dd` e estejam visíveis – `siblings('dd:visible')` e esconde-os por meio do efeito `slideUp()` a uma velocidade de 1,5 segundo. Fechamento da resposta que se revelou anteriormente..
Linha 13	Esta linha destina-se a repor o texto do link para ver a resposta que está sendo fechada. Experimente retirar essa linha, visualizar a página no navegador e constatar que para uma resposta fechada automaticamente, não há mais como revelá-la outra vez.

Sugestões:

- Faça uma cópia deste arquivo e experimente os efeitos `hide()`, `show()`, `fadeIn()` e `fadeOut()`, com diferentes velocidades. Depois, altere o disparador do evento de `click` para `mouseover`, `mouseout`, `mousemove`, `keydown`, `keyup` (estes dois últimos só para navegação por teclado).
- Nesta solução, não se mostra o link para o índice quando se revela a resposta. Modifique o script para mostrar o link. Dica: adicione, via script, classe(s) aos elementos HTML para viabilizar a solução. Por exemplo: use `$('dt + dd').addClass('resposta-oculta')`. Utilize o seletor de atributo `$('a[href="#indice"]')` para selecionar o link para o índice.

Experimente sempre, pois este é o melhor caminho para o aprendizado.

Quinta etapa

Nesta quinta etapa, será desenvolvida uma solução na qual o usuário, ao entrar na página da FAQ, encontrará somente a lista das perguntas e uma instrução orientando-o a clicar uma pergunta para revelar sua resposta.

O efeito final dessa solução pode ser visto na figura 9.7, onde constam, na parte superior, o estado inicial da página ao ser carregada e, na parte inferior, o resultado após o usuário ter clicado a primeira pergunta. O clique em outra pergunta fecha automaticamente a resposta anterior já revelada.

Figura 9.7 – Página para FAQ: quinta etapa.

Observe a seguir os códigos para essa solução.

▶ CSS:

```
.invisivel {display:none;}
.visivel-final {
   border:1px solid #c30;
   padding:5px 10px;
   background:#ff9;
   }
ol li a {
   color:blue;
   text-decoration:none;
   }
.cor-um {color:red;}
.cor-dois {color:blue;}
```

▶ **jQuery**:

```
1.   <script type="text/javascript">
2.       $(document).ready(function() {
3.           $('dt, dd').addClass('invisivel');
4.           $('ol').after('<p>Clique em uma pergunta para visualizar a resposta</p>');
5.           $('ol li a').click(function() {
6.               $('dt').css({display:'none'});
7.               $(this).removeClass('cor-dois').addClass('cor-um');
8.               $(this).parent().siblings().children().addClass('cor-dois');
9.               var idClicado = $(this).attr('href');
10.              $(idClicado + ' + dd').siblings('dd:visible').hide();
11.              $(idClicado).slideDown(1500);
12.              $(idClicado + ' + dd').fadeIn(2500).addClass('visivel-final');
13.              $(idClicado + ' + dd + dd').show();
14.          });
15.      });
16.  </script>
```

 [arquivo-9.1f.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Insere a classe `invisivel` em todos os elementos `dt` (este elemento contém a pergunta, logo acima na resposta) e `dd` (texto da resposta e link para índice logo abaixo da resposta).
Linha 4	Insere um parágrafo, logo após o índice, contendo um texto explicativo do funcionamento da página.
Linha 5	O script entrará em ação quando se clicar o texto de uma pergunta.
Linha 6	Retira a pergunta anterior quando se revela uma nova resposta. Sem esse método, a cada resposta revelada, as perguntas logo acima dela permanecerão no lugar, uma embaixo da outra. Experimente retirar esta linha e navegar nas respostas.
Linha 7	Atribui a classe `cor-um` (vermelha) à pergunta clicada, com a finalidade de destacá-la. Remove a classe `cor-dois` (azul) se presente no elemento.
Linha 8	Seleciona todos os elementos `a` no índice, que são irmãos do elemento clicado, e aplica neles a classe para a cor azul, garantindo que somente a pergunta clicada será destacada na cor vermelha. Para melhor entender a explicação para as linhas 7 a 9, considere que o usuário realizou um segundo clique no link. Se quiser tentar entender a lógica do script partindo do primeiro clique do usuário, talvez tenha dificuldades para compreendê-la.
Linha 9	Armazenou-se em uma variável o valor do atributo `href` do elemento `a` clicado (pergunta clicada).

Linha(s)	Descrição (cont.)
Linha 10	Aqui o seletor-irmão adjacente (seletores do tipo `E + F`) - `$(idClicado + ' + dd')` seleciona o elemento `dd` que se segue imediatamente ao elemento cuja `id` é o valor da variável `idClicado`. Se você acompanhar atentamente a marcação HTML, constatará que se selecionou a resposta à pergunta clicada. A seguir, escolheu-se, entre os irmãos desse elemento, aquele que se encontra visível (`dd:visible`); trata-se da resposta revelada que se escondeu. Resumindo: esta linha esconde a resposta revelada quando o usuário clica para revelar outra resposta.
Linha 11	Revela a pergunta logo acima da resposta em um efeito de movimento vertical de cima para baixo (`slideDown()`).
Linha 12	Revela e estiliza a resposta à pergunta clicada.
Linha 13	Revela o link para o índice logo abaixo da resposta.

Com a conclusão dessa quinta etapa, finalizou-se a explicação dos fundamentos básicos para revelar e esconder conteúdos de uma FAQ com o uso de jQuery. Utilizou-se uma FAQ CSS como exemplo para desenvolver os códigos, mas os conceitos se aplicam a qualquer tipo de conteúdo, incluindo menus e imagens. É importante que você tenha entendido a lógica e o funcionamento de cada script mostrado nessas cinco etapas.

Os exemplos mostrados foram desenvolvidos para uma marcação HTML em uma página sem outro tipo de conteúdo. Em um site real, a FAQ ou outro conteúdo qualquer no qual se aplicaria jQuery estariam inseridos em uma página com colunas de navegação, topo, rodapé etc., com seus divs, parágrafos, `ul`, `ol`, `dt`, `dl`, `dd`, `h1`, `h2` e uma série de outros elementos HTML que compõe uma página web. Os seletores jQuery devem selecionar somente os elementos envolvidos nos conteúdos a animar. Assim, se a FAQ em questão estivesse inserida em uma página web, seria necessário prever mecanismos para que o script não selecionasse elementos fora da marcação da FAQ.

Uma solução seria criar um container para a área da marcação HTML onde será aplicado o script e fazer referência a esse container, como elemento-raiz para todos os seletores jQuery. Observe os códigos a seguir.

▶ **HTML**:

```
...
<div id="topo"><p>Topo do site</p></div>
<div id="principal">
<h1>Título da página</h1>
<p>Parágrafo</p>
...mais conteúdo...
<h2>FAQ CSS</h2>

<div id="faq">
... HTML à aplicar jQuery ...
</div> <!-- Fim do div#faq -->

...mais conteúdo...
</div> <!-- Fim do div#principal -->
<h4>Navegação</h4>
<li>Link 1</li>
...
<li>Link 10</li>
</div>
...
```

Os seletores jQuery teriam uma sintaxe como a mostrada a seguir:

```
$('#faq p');
$('#faq a.ver').click(function() {
// script aqui
});
$('#faq' + idClicado + ' + dd');
$('#faq dt');
```

9.2 Página de notícias

Para encerrar o assunto proposto neste capítulo, você irá estudar um caso de aplicação prática bem mais abrangente que a FAQ. Praticamente todos os blogs e muitos sites utilizam a técnica de apresentar o título de uma matéria, seguido de um pequeno trecho descritivo do conteúdo da matéria e um link do tipo "Saiba mais...", convidando o usuário a clicar o link e ler a matéria.

Inspirando-se nessa forma de apresentação de matéria ou notícia, você irá aplicar o que aprendeu sobre revelar e esconder conteúdos, em uma página de notícias, aproveitando para estudar o seletor de negação, ou filtro de negação jQuery, o loop com o método `each()` e alguma manipulação de arrays.

Marcação básica

Tal como fez para a FAQ, você utilizará uma marcação HTML básica na qual consta um total de quatro notícias. O script será desenvolvido de modo a ser aplicável a qualquer número de notícias. Considere uma inserção dinâmica de notícias.

A renderização da página básica de notícias, para a qual o script será desenvolvido, é mostrado na figura 9.8.

Figura 9.8 – Página básica para notícias.

O documento em questão encontra-se em sua forma mais elementar, com alguma estilização básica. Conforme se pode observar na figura 9.8, as notícias vão sendo inseridas com seu texto completo, podendo conter figuras ou outro tipo qualquer de conteúdo. Ainda que o usuário desabilite suporte às CSS em seu navegador ou abra a página em um navegador de texto, terá acesso a todas as notícias e nenhum conteúdo será perdido.

A folha de estilo e a marcação HTML da página básica para os exemplos são mostradas a seguir.

▶ **CSS**:

```
1.    <style type="text/css" media="all">
2.    body {width:400px; font:12px/1.2 Arial, Helvetica, sans-serif; margin:10px
      auto 0 auto; padding:0;}
3.    h1 {width:415px; color:#c60; font-size:20px; padding-left:5px; border-
      bottom:1px solid #c60;}
```

```
19            $('.sm').text('Saiba mais...');
20          });
21.       });
22.    </script>
```

 [arquivo-9.2b.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Analise este seletor por partes. O sinal estrela (*) identifica o seletor universal, que encontra todos os elementos HTML. É o seu conhecido curinga. A sintaxe `*:not(h2.tit)` significa o seguinte: todos os elementos menos os cabeçalhos `h2` com a classe `tit`. Então, `.noticias *:not(h2.tit)` é um seletor composto que seleciona tudo, dentro dos divs com a classe `noticias`, menos o título com a classe `tit`, e esconde `hide()`. Resumindo: todos os conteúdos serão escondidos menos seus títulos.
Linha 4	Seleciona os títulos das notícias e insere dentro deles um link denominado "Saiba mais..." com a classe `sm`. O método `append()` insere marcação HTML em um container existente. Esta linha resulta na seguinte marcação: `<h2>Notícia xxx  <a href="#" class="sm">Saiba mais...</a></h2>`. Criou-se um link dentro do título da notícia. A renderização sem estilização para essa marcação faria o texto do link ser renderizado ao lado do texto do título e em fonte do tamanho e cor da do título. Alteraram-se a posição, a cor e o tamanho do link com declarações CSS para a classe `sm`. Link dentro de um cabeçalho? Uma agressão à marcação em conformidade com os padrões? Optou-se por essa solução, propositadamente, com a finalidade de alertar quanto à escolha dos métodos jQuery. O script em questão irá funcionar sem problemas, mas aos olhos atentos de um desenvolvedor consciente dos Padrões Web essa marcação seria reprovada. Ou não? Excesso de preciosismo? Alguém poderia argumentar que a marcação extra não aparece no código da página. Cabe a você decidir o que fazer. Como sugestão, utilize um outro método jQuery no lugar de `append()`. O que você acha de `after()`?
Linha 5	Insira o link para fechar uma notícia aberta. A inserção se faz dentro do `div` que contém a notícia logo em seu início `prepend()`, pois se deseja o link em cima. Se tivesse usado `append()`, o link teria ficado debaixo do `div`. Seria melhor embaixo? Talvez sim. O leitor chegaria ao final da notícia e não haveria necessidade de rolar de volta para cima com a finalidade de encontrar o link. Utilizou-se `prepend()` com o objetivo de destacar a diferença para `append()`.
Linha 6	Início da ação quando o usuário clica o link "Saiba mais...".

Linha(s)	Descrição (cont.)
Linha 7	Já comentado anteriormente. Mesmo efeito de `return false` da linguagem JavaScript.
Linha 8	Entre todos os elementos ancestrais do link clicado, encontre aquele cuja classe é `noticias`. Selecione todos os seus filhos e revele-os. Acompanhe na marcação. Os elementos ancestrais do link são, em ordem do mais próximo para o mais distante, `h2` (o link está dentro de um `h2`), `div` com a classe `noticias`, `body` e, finalmente, `html`. Resumindo: será revelado todo o conteúdo dentro do `div` onde se encontra o link clicado.
Linha 9	Coloca uma cor de fundo diferente (verde) no `div` da notícia revelada.
Linha 10	Esta linha só causa efeito após o primeiro clique. Destina-se a fechar automaticamente uma notícia revelada quando outra recebe um clique para revelação.
Linha 11	Devolve a cor de fundo para um `div` que acaba de ser fechado. Note que essa cor `#cff` (azul-clara) é a mesma declarada nas CSS (linha 8 das CSS).
Linha 12	Devolve o link "Saiba mais..." para um `div` que acaba de ser fechado.
Linha 13	Retira o link "Saiba mais..." do `div` que acaba de ser revelado.
Linha 15	Início da ação quando o usuário clica o link "Fechar".
Linha 16	Comentado anteriormente. Mesmo efeito de `return false` da linguagem JavaScript.
Linha 17	Esconde todo o conteúdo do `div` menos o título da notícia e o link. Note o seletor de negação.
Linha 18	Devolve a cor azul ao `div` fechado.
Linha 19	Devolve o link "Saiba mais..." ao `div` fechado.

Segunda etapa

O objetivo para esta segunda etapa é fornecer o trecho do texto que inicia a notícia com o link "Saiba Mais...".

Esta solução está sendo proposta com a finalidade de estudar o uso de interação e recuperação de elementos de arrays. Veja a seguir um maneira de manipular arrays com a biblioteca jQuery.

Observe, na figura 9.10, o resultado final dessa solução.

Figura 9.10 – Página para notícias: segunda etapa.

Para capturar o trecho de texto que irá aparecer inicialmente antes do link "Saiba mais...", marque-o no HTML da página usando o elemento neutro `span`. Tal marcação é dispensável, pois você pode capturar o texto com jQuery. Essa opção de captura é um exercício para você testar seu aprendizado.

Na marcação, cada notícia recebe um elemento `span` no início do primeiro parágrafo como mostrado a seguir. O conteúdo desse elemento irá aparecer antes do link "Saiba mais...".

▶ **HTML**:

```
...
<h1>Notícias</h1>
<div class="noticias">
<h2 class="tit">Notícia um</h2>
<p><span>Texto da primeira notícia. Lorem ipsum dolor sit amet, consectetuer
   adipiscing elit.</span>
Morbi eleifend, purus ...</p>
...
```

O desafio é identificar o conteúdo de cada elemento `span`, capturar e escrever, como mostra a figura 9.10.

A seguir, o script jQuery:

```
1.    <script type="text/javascript">
2.       $(document).ready(function() {
```

CAPÍTULO 10

Efeitos em tabelas

Neste capítulo, serão abordados alguns efeitos da biblioteca jQuery destinados a incrementar a apresentação de tabelas de dados em documentos para web. Destaca-se que, também para tabelas, continua válido o conceito de JavaScript não obstrutivo.

10.1 Destinação das tabelas HTML

Segundo os fundamentos dos Padrões Web, as tabelas HTML se destinam à apresentação de dados tabulares. Veja alguns exemplos de dados tabulares: pesos, medidas, índices financeiros, preços, resultados de competições etc.

É equivocado o conceito, disseminado principalmente por aqueles que se iniciam no estudo dos Padrões Web, de que as tabelas são proibidas na marcação HTML. Ao contrário, tabelas são previstas pelo W3C, são legais, e devem ser usadas para apresentar dados.

Tabelas não são admitidas na construção de layout ou para obtenção de efeitos de apresentação. Essas tarefas são de responsabilidade exclusiva das CSS.

10.2 Marcação de tabelas

As Recomendações do W3C para a HTML preveem os seguintes elementos para marcação de tabelas: `table`, `caption`, `thead`, `tfoot`, `tbody`, `colgroup`, `col`, `tr`, `th` e `td`.

São dez elementos dos quais `colgroup` e `col` são de uso específico e `tfoot` destina-se a tabelas que contenham um rodapé. Assim, restam oito elementos que deveriam estar presentes obrigatoriamente na marcação de qualquer tabela, por mais simples que esta seja.

Quantos desses elementos você usa na marcação de suas tabelas? A maioria das tabelas encontradas na web usa os elementos `table`, `tr` e `td`. Se você é um dos que usam somente esses três elementos, estude e repense seus conceitos de construção de tabelas. Não é do escopo deste livro aprofundar esse assunto, contudo, para as tabelas deste capítulo, será utilizada a marcação segundo as Recomendações do W3C.

10.3 Tabela de horários de ônibus

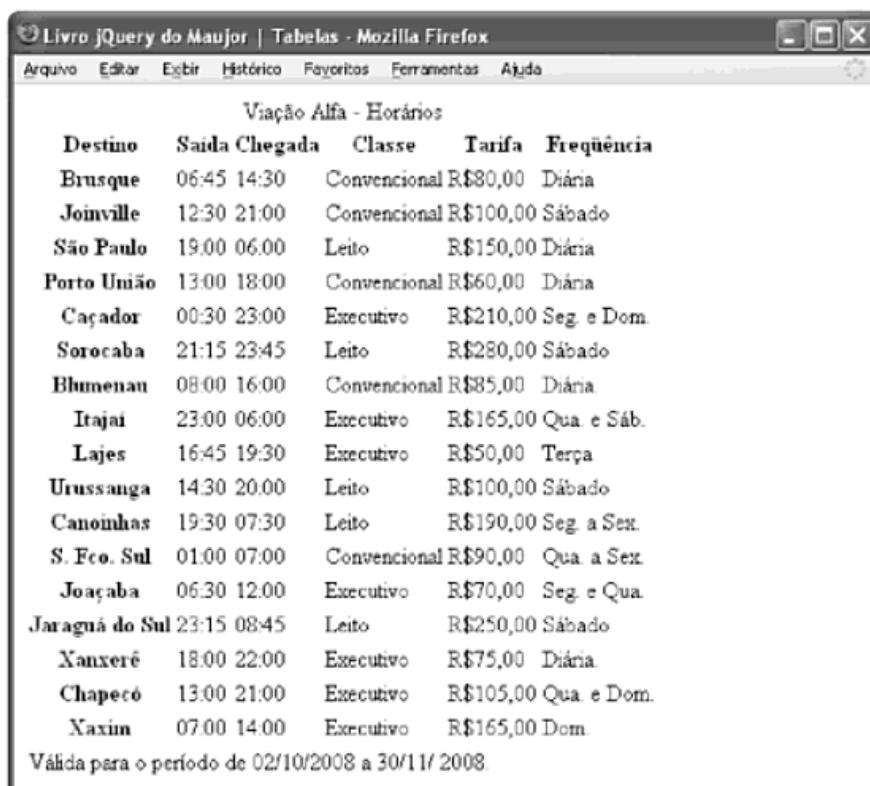
Para explicar os efeitos propostos para este capítulo, será utilizada uma tabela publicada por uma empresa de ônibus fictícia, denominada Viação Alfa, destinada a apresentar os dados tabulares referentes aos horários de partida e chegada de seus ônibus, bem como a classe do ônibus e o preço da passagem. Note ainda que não houve preocupação com a apresentação de dados coerentes na tabela, pois o objetivo é mostrar as técnicas de criação de efeitos, e não comparar preços de passagem ou duração de viagem.

Marcação básica

Veja a seguir a marcação básica da tabela em questão. Antes de prosseguir com seus estudos, procure acompanhar com detalhes a marcação HTML e estilização, com a finalidade de facilitar o entendimento dos scripts desenvolvidos.

▶ **HTML**:

```
1.      <table id="horario">
2.      <caption>Viação Alfa - Horários</caption>
3.      <thead>
4.          <tr id="horizontal"> <!-- linha do cabeçalho -->
5.              <th>Destino</th>
6.              <th scope="col">Saída</th>
7.              <th scope="col">Chegada</th>
8.              <th scope="col">Classe </th>
9.              <th scope="col">Tarifa </th>
10.             <th scope="col">Freq&uuml;ência</th>
11.         </tr>
12.     </thead>
13.     <tfoot>
14.         <tr> <!-- linha do rodapé -->
15.             <td colspan="6">Válida para o período de 02/10/2008 a 30/11/ 2008.</td>
16.         </tr>
17.     </tfoot>
```

```
18.     <tbody>
19.         <tr> <!-- primeira linha de dados da tabela -->
20.             <th scope="row">Brusque</th>
21.             <td>06:45</td>
22.             <td>14:30</td>
23.             <td>Convencional</td>
24.             <td>R$80,00</td>
25.             <td>Diária</td>
26.         </tr>
27.         ... marcação idêntica para quinze linhas intermediárias da tabela ...
28.         <tr> <!-- última linha de dados da tabela -->
29.             <th scope="row">Xaxim</th>
30.             <td>07:00</td>
31.             <td>14:00</td>
32.             <td>Executivo</td>
33.             <td>R$165,00</td>
34.             <td>Dom.</td>
35.         </tr>
36.     </tbody>
37.     </table>
```

 [arquivo-10.3a.html]

Código comentado:

Linha(s)	Descrição
Linha 1	Atribuiu-se um identificador id para a tabela em questão.
Linha 2	Uso do elemento caption para fornecer uma descrição da natureza da tabela. Em tabelas mais complexas, essa descrição deverá ser complementada com o uso do atributo summary no elemento table. O atributo summary destina-se a informar a finalidade da tabela e fornecer uma descrição de sua estrutura.
Linhas 3 e 12	O elemento thead destina-se a marcar as linhas que compõem o cabeçalho da tabela. Neste caso, a tabela contém uma linha de cabeçalho. Esse elemento é o seletor natural para o cabeçalho da tabela. A maioria dos desenvolvedores não usa, erroneamente, esse elemento.
Linhas 4 e 11	Elemento tr para marcação da linha de cabeçalho.
Linhas 5 a 10	Marcação das células do cabeçalho com o uso do elemento th. A maioria dos desenvolvedores usa, erroneamente, o elemento td para marcar células de cabeçalho. Note a presença do atributo scope para fins de acessibilidade.

Linha(s)	Descrição (cont.)
Linhas 13 a 17	Marcação do rodapé da tabela com o uso do elemento `tfoot`. Parece estranho o rodapé vir logo após o cabeçalho, mas é isso mesmo. As Recomendações do W3C para a HTML preveem que o cabeçalho e o rodapé sejam carregados na página antes do corpo da tabela que deverá ser renderizada na seguinte ordem: cabeçalho, rodapé e corpo. Isto é muito conveniente em tabelas com muitas linhas que demoram a carregar. O usuário visualiza o cabeçalho e o rodapé imediatamente enquanto as linhas vão sendo carregadas. Esse elemento é o seletor natural para o rodapé da tabela. A maioria dos desenvolvedores não usa, erroneamente, esse elemento.
Linhas 18 e 36	O elemento `tbody` destina-se a marcar as linhas que compõem o corpo da tabela. Neste caso, a tabela contém dezessete linhas em seu corpo, das quais somente a primeira e a última foram mostradas na marcação. Esse elemento é o seletor natural para o corpo da tabela. A maioria dos desenvolvedores não usa, erroneamente, esse elemento.
Linhas 19 a 35	Marcação das linhas do corpo da tabela. Note a presença do atributo `scope` na primeira célula de cada linha para fins de acessibilidade.

Na figura 10.1, mostramos a renderização da tabela segundo a marcação básica proposta.

Viação Alfa - Horários

Destino	Saída	Chegada	Classe	Tarifa	Freqüência
Brusque	06:45	14:30	Convencional	R$80,00	Diária
Joinville	12:30	21:00	Convencional	R$100,00	Sábado
São Paulo	19:00	06:00	Leito	R$150,00	Diária
Porto União	13:00	18:00	Convencional	R$60,00	Diária
Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	Leito	R$280,00	Sábado
Blumenau	08:00	16:00	Convencional	R$85,00	Diária
Itajaí	23:00	06:00	Executivo	R$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R$50,00	Terça
Urussanga	14:30	20:00	Leito	R$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R$90,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
Xanxerê	18:00	22:00	Executivo	R$75,00	Diária
Chapecó	13:00	21:00	Executivo	R$105,00	Qua. e Dom.
Xaxim	07:00	14:00	Executivo	R$165,00	Dom.

Válida para o período de 02/10/2008 a 30/11/ 2008.

Figura 10.1 – Tabela sem estilização.

Criou-se uma folha de estilos para a tabela em questão, mostrada a seguir.

▶ **CSS**:

```
<style type="text/css" media="all">
body {
	width:600px;
	font:80%/1.2 Arial, Helvetica, sans-serif;
	margin:30px auto;
	padding:0;
	color:#666;
	}
table {
	width:550px;
	border-collapse:collapse;
	border:2px solid #999;
	margin:0 auto;
	}
caption {
	text-align:right;
	margin-bottom:0.3em;
	border-bottom:1px solid #333;
	padding-right:0.3em;
}
thead tr th {
	text-align:center;
	border-bottom:2px solid #999;
	border-left:2px solid #999;
	}
tr td, tr th {
	padding:1px 5px;
	text-align:left;
	font-size:0.9em;
	border:1px dotted #333;
	}
tfoot tr td {
	text-align:center;
	border-top:2px solid #999;
	}
</style>
```

[arquivo-10.3b.html]

Trata-se de uma folha de estilos mínima destinada a centralizar a página e inserir apresentação básica com a finalidade exclusiva de facilitar a visualização do exemplo.

Zebra par-ímpar

É a tradicional estilização de linhas pares com uma determinada cor de fundo e ímpares com outra cor. O primeiro efeito consiste em simplesmente atribuir uma cor de fundo a uma das linhas (par ou ímpar) e deixar a cor de fundo da outra com a cor-padrão do fundo da tabela.

Na figura 10.3, veja o efeito final de uma estilização zebra par-ímpar, obtida com jQuery, com uma das cores sendo a cor-padrão da tabela.

Livro jQuery do Maujor | Tabelas - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

Viação Alfa - Horários

Destino	Saída	Chegada	Classe	Tarifa	Freqüência
Brusque	06:45	14:30	Convencional	R$80,00	Diária
Joinville	12:30	21:00	Convencional	R$100,00	Sábado
São Paulo	19:00	06:00	Leito	R$150,00	Diária
Porto União	13:00	18:00	Convencional	R$60,00	Diária
Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	Leito	R$280,00	Sábado
Blumenau	08:00	16:00	Convencional	R$85,00	Diária
Itajaí	23:00	06:00	Executivo	R$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R$50,00	Terça
Urussanga	14:30	20:00	Leito	R$100,00	Sábado
Canoinhas	13:30	07:30	Leito	R$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R$90,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
Xanxerê	18:00	22:00	Executivo	R$75,00	Diária.
Chapecó	13:00	21:00	Executivo	R$105,00	Qua. e Dom.
Xaxim	07:00	14:00	Executivo	R$165,00	Dom.
Valida para o período de 02/10/2008 a 30/11/2008					

Figura 10.3 – Efeito zebra par-ímpar, uma cor-padrão.

A biblioteca jQuery prevê os pseudosseletores `:even` e `:odd` que se destinam a selecionar as ocorrências, respectivamente, pares e ímpares do conjunto de elementos-alvo do seletor, qualquer que seja ele.

Assim, se você escrever o seletor `$('table#horario tbody tr:odd')`, estará selecionando todas as linhas ímpares contidas no corpo da tabela cujo `id` é *`horario`*.

Agora observe o script a seguir:

```
<script type="text/javascript">
   $(document).ready(function() {
      $('table#horario tbody tr:odd').addClass('impar');
   });
</script>
```

[arquivo-10.4a.html]

Os parâmetros `odd` ou `even` selecionam os elementos-filho ímpares ou pares respectivamente. Ao contrário dos pseudosseletores `:odd` e `:even` estudados anteriormente, as ocorrências pares são a segunda, quarta etc., pois a contagem para o índice não se baseia em contagem JavaScript e, assim, começa em um e não zero. Desta forma, `tr:nth-child(odd)` seleciona as linhas ímpares de uma tabela, enquanto `tr:odd`, as linhas pares.

O parâmetro *`equação`* destina-se a selecionar elementos-filho em uma posição definida pelo resultado de uma equação. A forma geral da equação é mostrada a seguir:

```
x = an + b
```

Onde `x` é a ordem numérica onde se encontra o elemento-filho, `a` e `b` são números naturais e `n` são os números do conjunto {0, 1, 2, 3...}.

Por exemplo:

Equação	Seleção
2n	Seleciona as ocorrências 2x0 = 0, 2x1 = 2, 2x2 = 4, 2x3 = 6,..., ou seja, ocorrências pares.
2n + 1	Ocorrências 2x0 + 1 = 1, 2x1 + 1 = 3, 2x2 + 1 = 5, 2x3 + 1 = 7,..., ou seja, ocorrências ímpares.
3n + 1	Ocorrências 3x0 + 1 = 1, 3x1 + 1 = 4, 3x2 + 1 = 7, 3x3 + 1 = 10,..., ou seja, ocorrências de três em três.
5n - 1	Ocorrências 5x0 - 1 = -1, 5x1 - 1 = 4, 5x2 - 1 = 9, 5x3 - 1 = 14,..., ou seja, ocorrências de cinco em cinco. Aqui, a primeira ocorrência resultou em um número negativo. Números negativos como resultados da equação são ignorados para efeito de seleção.

É fácil concluir que o uso de uma equação para selecionar ocorrências aumenta muito as possibilidades de seleção e combinações.

Efeitos zebra mais complexos e diferentes dos estudados até aqui podem requerer o uso de uma equação combinada com os métodos `prev()` ou `next()` destinados a selecionar elementos antes e depois, respectivamente, do elemento selecionado. Por exemplo:

Seletor	Seleção
`$('tr:nth-child(4)`	Seleciona a quarta linha da coluna.
`$('tr:nth-child(4).prev()`	Seleciona a terceira linha da coluna.
`$('tr:nth-child(4).next()`	Seleciona a quinta linha da coluna.

Com a finalidade de verificar seu aprendizado, antes de prosseguir a leitura, faça uma cópia do `arquivo-10.3b.html` e, nele, escreva o script para obter o efeito zebra dois-dois mostrado na figura 10.5.

Livro jQuery do Maujor | Tabelas - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

Viação Alfa - Horários

Destino	Saída	Chegada	Classe	Tarifa	Freqüência
Brusque	06:45	14:30	Convencional	R$80,00	Diária
Joinville	12:30	21:00	Convencional	R$100,00	Sábado
São Joaquim	19:00	06:00	Leito	R$150,00	Diária
Porto União	13:00	18:00	Convencional	R$80,00	Diária
Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
Sombrio	21:15	13:45	Leito	R$280,00	Sábado
Blumenau	08:00	16:00	Convencional	R$85,00	Diária
Itajaí	23:00	03:00	Executivo	R$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R$50,00	Terça
Urussanga	14:30	20:00	Leito	R$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R$90,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
Xanxeré	18:00	22:00	Executivo	R$75,00	Diária
Chapecó	13:00	21:00	Executivo	R$105,00	Qua. e Dom.
Xaxim	07:00	14:00	Executivo	R$165,00	Dom.
Válida para o período de 02/10/2008 a 30/11/2008.					

Figura 10.5 – Efeito zebra dois-dois.

Observe que as linhas a estilizar com cor de fundo diferente da cor-padrão são as linhas: 3, 7, 11, 15... e as linhas que se seguem a cada uma delas: 4, 8, 12, 16... ou, dito de outra forma, as linhas: 4, 8, 12, 16... e as linhas que antecedem cada uma delas: 3, 7, 11, 15...

Assim, há duas opções de script. Em ambas as opções, a série de números a selecionar dá um salto de quatro unidades. No exemplo em questão, será adotada a segunda opção na qual os números são múltiplos de quatro.

O script é o seguinte:

```
<script type="text/javascript">
   $(document).ready(function() {
      /* seleciona os múltiplos de quarto */
      $('table#horario tbody tr:nth-child(4n)').addClass('cor_um');
      /* seleciona os anteriores aos múltiplos de quarto */
      $('table#horario tbody tr:nth-child(4n)').prev().addClass('cor_um');
   });
</script>
```

A classe `.cor-um` recebe a regra de estilo mostrada a seguir:

```
.cor_um {background:#bfe2d8;}
```

Progressão aritmética

Séries de números nas quais cada número é igual a seu precedente mais um valor constante são denominadas de progressão aritmética (PA). A matemática ensina que o primeiro termo de uma PA denomina-se termo inicial, o valor constante é a razão da PA é a fórmula do termo geral é:

```
an = (n-1)*r + a1
```

Onde `an` é o termo de ordem `n`, `a1` é o primeiro termo, `r` é a razão e `n` é a ordem do termo.

Aplicando a fórmula na primeira PA 3, 7, 11..., você encontrará:

an = (n-1)*4 + 3 => 4n - 4 + 3 => 4n - 1

Aplicando a fórmula na segunda PA 4, 8, 12..., você encontrará:

an = (n-1)*4 + 4 => 4n - 4 + 4 => 4n

Assim, com o uso da fórmula das PAs, há uma forma mais genérica de seleção sem a necessidade de emprego dos métodos `prev()` e `next()`, e o script em questão tem a seguinte sintaxe:

```
<script type="text/javascript">
   $(document).ready(function() {
      $('table#horario tbody tr:nth-child(4n-1)').addClass('cor_um');
      $('table#horario tbody tr:nth-child(4n)').addClass('cor_um');
   });
</script>
```

 [arquivo-10.4c.html]

Zebra três cores

A técnica para este efeito é a mesma para o efeito anterior. Desenvolva tal solução adotando três cores diferentes da cor-padrão e, assim, sua folha de estilo contemplará três classes a serem inseridas por script. As regras de estilo para essas classes são mostradas a seguir:

```
.cor_um {background:#d6e2e5;}      /* cor azul mais clara que a da cor três */
.cor_dois {background:#bfe2d8;}    /* cor verde-clara */
.cor_tres {background:#add6ef;}    /* cor azul-clara */
```

Observe, na figura 10.6, o efeito final com três cores alternadas.

Livro jQuery do Maujor | Tabelas - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

Viação Alfa - Horários

	Destino	Saída	Chegada	Classe	Tarifa	Frequência
1	Brusque	06:45	14:30	Convencional	R$80,00	Diária
2	Joinville	12:30	21:00	Convencional	R$100,00	Sábado
3	São Joaquim	19:00	08:00	Leito	R$150,00	Diária
4	Porto União	13:00	18:00	Convencional	R$60,00	Diária
5	Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
6	Sombrio	21:15	13:45	Leito	R$280,00	Sábado
7	Blumenau	08:00	16:00	Convencional	R$85,00	Diária.
8	Itajaí	23:00	03:00	Executivo	R$165,00	Qua. e Sáb.
9	Lajes	16:45	19:30	Executivo	R$50,00	Terça
10	Urussanga	14:30	20:00	Leito	R$100,00	Sábado
11	Canoinhas	18:30	07:30	Leito	R$190,00	Seg. a Sex.
12	S. Fco. Sul	01:00	07:00	Convencional	R$90,00	Qua. a Sex.
13	Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
14	Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
15	Xanxerê	18:00	22:00	Executivo	R$75,00	Diária.
16	Chapecó	13:00	21:00	Executivo	R$105,00	Qua. e Dom.
17	Xaxim	07:00	14:00	Executivo	R$165,00	Dom.
	Válida para o período de 02/10/2008 a 30/11/2008.					

Figura 10.6 – Efeito zebra três cores.

Note, na figura 10.6, que as três cores de fundo se distribuem pelas linhas da tabela conforme mostrado a seguir.

Cor	Linhas
cor-um	1,4,7,10,13,16. PA de razão igual a 3 e a1 igual a 1.
cor-dois	2,5,8,11,14,17. PA de razão igual a 3 e a1 igual a 2.
cor-tres	3,6,9,12,15. PA de razão igual a 3 e a1 igual a 3.

E os seletores são:

Cor	Seletor
cor-um	(n-1)*3 + 1 => 3n-3+1 => 3n-2.
cor-dois	(n-1)*3 + 2 => 3n-3+2 => 3n-1.
cor-tres	(n-1)*3 + 3 => 3n-3+3 => 3n.

Finalmente, o script para o efeito zebra três cores:

```
<script type="text/javascript">
   $(document).ready(function() {
      $('table#horario tbody tr:nth-child(3n-2)').addClass('cor_um');
      $('table#horario tbody tr:nth-child(3n-1)').addClass('cor_dois');
      $('table#horario tbody tr:nth-child(3n)').addClass('cor_tres');
   });
</script>
```

[arquivo-10.4d.html]

Veja os seletores a seguir:

Cor	Seletor
cor-um	(n-1)*6 + 1 => 6n-6+1 => 6n-5. (primeira PA).
cor-um	(n-1)*6 + 2 => 6n-6+2 => 6n-4. (segunda PA).
cor-um	(n-1)*6 + 3 => 6n-6+3 => 6n-3. (terceira PA).
cor-dois	(n-1)*6 + 4 => 6n-6+4 => 6n-2. (primeira PA).
cor-dois	(n-1)*6 + 5 => 6n-6+5 => 6n-1. (segunda PA).
cor-dois	(n-1)*6 + 6 => 6n-6+6 => 6n. (terceira PA).

E o script para o efeito zebra três-três:

```
<script type="text/javascript">
   $(document).ready(function() {
      $('table#horario tbody tr:nth-child(6n-5)').addClass('cor_um');
      $('table#horario tbody tr:nth-child(6n-4)').addClass('cor_um');
      $('table#horario tbody tr:nth-child(6n-3)').addClass('cor_um');
      $('table#horario tbody tr:nth-child(6n-2)').addClass('cor_dois');
      $('table#horario tbody tr:nth-child(6n-1)').addClass('cor_dois');
      $('table#horario tbody tr:nth-child(6n)').addClass('cor_dois');
   });
</script>
```

Uma outra solução para obter esse efeito faz uso dos métodos `prev()` e `next()` para selecionar linhas antes e depois da linha selecionada, conforme se mencionou anteriormente.

Observe que as linhas 2, 8 e 14 são as linhas centrais de cada grupo de três linhas que têm a cor de fundo definida pela classe `cor-um`. O seletor que adiciona a classe nessas linhas centrais é:

```
$('table#horario tbody tr:nth-child(6n-4)').addClass('cor_um');
```

De modo semelhante, para as linhas centrais que têm a cor de fundo definida pela classe `cor-dois`, o seletor é:

```
$('table#horario tbody tr:nth-child(6n-1)').addClass('cor_dois');
```

Tendo adicionado a classe correspondente à linha central de cada grupamento de três linhas, o seletor para a linha anterior é:

```
$('table#horario tbody tr:nth-child(6n-4)').prev().addClass('cor_um');
$('table#horario tbody tr:nth-child(6n-1)').prev().addClass('cor_dois');
```

E o seletor para a linha posterior é:

```
$('table#horario tbody tr:nth-child(6n-4)').next().addClass('cor_um');
$('table#horario tbody tr:nth-child(6n-1)').next().addClass('cor_dois');
```

Use a tabela desenvolvida no `arquivo-10.4b.html` (Figura 10.4) para obter esse efeito. O script é mostrado a seguir:

```
1.     <script type="text/javascript">
2.     $(document).ready(function() {
3.        $('table#horario tbody tr:odd').addClass('impar');
4.        $('table#horario tbody tr:even').addClass('par');
5.        $('table#horario tbody tr').hover(
6.           function() {
7.              $(this).addClass('destacar');
8.           },
9.           function() {
10.             $(this).removeClass('destacar');
11.          }
12.       );
13.    });
14.    </script>
```

[arquivo-10.5a.html]

Código comentado:

Linha(s)	Descrição
Linhas 3 e 4	Efeito zebra estudado anteriormente.
Linha 5	Atrelando o efeito `hover()` a cada uma das células da tabela.
Linhas 6 a 8	Função a ser executada quando o ponteiro do mouse for passado sobre a linha.
Linha 7	Adiciona a classe `destacar` à linha que recebeu o ponteiro do mouse.
Linhas 9 e 11	Função a ser executada quando o ponteiro do mouse abandonar a linha.
Linha 10	Remove a classe `destacar` da linha da qual o ponteiro do mouse foi retirado.
Linha 12	Fim do método `hover()`.

Destacar colunas

Agora, você estudará uma técnica para destacar as colunas da tabela. Informará ao usuário que é preciso que ele clique uma determinada célula de cabeçalho para destacar a coluna correspondente.

Observe, na figura 10.9, o efeito final para destacar a coluna denominada `Classe` após o usuário ter clicado a sua célula de cabeçalho.

Destino	♦ Saída	♦ Chegada	♦ Classe	♦ Tarifa	♦ Frequência
Brusque	06:45	14:30	Convencional	R$80,00	Diária
Joinville	12:30	21:00	Convencional	R$100,00	Sábado
São Paulo	19:00	06:00	Leito	R$150,00	Diária
Porto União	13:00	18:00	Convencional	R$60,00	Diária
Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	Leito	R$150,00	Sábado
Blumenau	08:00	16:00	Convencional	R$100,00	Diária
Itajaí	23:00	06:00	Executivo	R$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R$80,00	Terça
Urussanga	14:30	20:00	Leito	R$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R$80,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
Xanxerê	18:00	22:00	Executivo	R$60,00	Seg. a Sex.
Chapecó	13:00	21:00	Executivo	R$105,00	Seg. e Dom.
Xaxim	07:00	14:00	Executivo	R$165,00	Dom.
Válida para o período de 02/10/2008 a 30/11/ 2008.					

Figura 10.9 – Destacar coluna.

Use a tabela desenvolvida no `arquivo-10.4b.html` (Figura 10.4) para exemplificar esse efeito. As regras de estilo para a classe `destacar` são idênticas às adotadas no exemplo anterior. O script é mostrado a seguir:

```
1.  <script type="text/javascript">
2.  $(document).ready(function() {
3.      $('table#horario tbody tr:odd').addClass('impar');
4.      $('table#horario tbody tr:even').addClass('par');
5.      $('table#horario tr#horizontal th').not('th:first-child')
6.      .prepend('<span>&diams; </span>').css('cursor', 'pointer');
7.          $('table#horario tr#horizontal th').each(function(i) {
8.              var n = i - 1;
9.              $(this).click(function() {
10.                 $('td').removeClass('destacar');
11.                 $(this).parents('thead')
12.                 .siblings('tbody')
13.                 .children('tr').each(function() {
14.                     $(this).children('td:eq(' + n + ')').addClass('destacar');
15.                 });
16.             });
17.         });
18.     });
19. </script>
```

[arquivo-10.5b.html]

Linha(s)	Descrição (cont.)
Linha 14	Para cada linha da tabela, procura o elemento-filho que seja uma célula de dados `td` e seleciona aquela que se encontra na posição n. A variável `n` foi definida na linha 8 do script e vale i-1. Acompanhe: o usuário clicou a coluna cujo cabeçalho é "Classe", i=3 (trata-se da quarta coluna da tabela) e n=2 (i-1). Assim, para cada linha da tabela, o script seleciona a `td` de ordem 2 (valor de n), ou seja, a terceira célula de dados da linha e a esta adiciona a classe para destacar. Como a interação é por todas as linhas do corpo da tabela, todas as terceiras células de dados receberão a classe para destacar, criando o efeito de destaque em toda a coluna. Note que a primeira célula de cada linha é uma célula de cabeçalho `th` e não uma célula de dados `td`. Se a primeira coluna não fosse uma coluna de cabeçalhos e você quisesse destacá-la como as demais, usaria a variável `i` como índice, sem necessidade de criar a variável `n`.

Destacar linhas seletivamente

Denomina-se efeito destacar linhas seletivamente aquele segundo em que o usuário clica uma célula da tabela e todas as linhas que contêm o dado igual àquele da célula clicada são destacadas.

Primeiramente, informe o usuário de que ele deverá passar o ponteiro do mouse sobre qualquer célula de dados. Essa ação fará aparecer uma dica (tooltip) informando ao usuário que se ele clicar a célula, as linhas que contiverem esse dado serão destacadas. Esta primeira etapa é apresentada na figura 10.10.

Destino	Saída	Chegada	Classe	Tarifa	Freqüência
Brusque	06:45	14:30	Convencional	R$80,00	Diária
Joinville	12:30	21:00	Convencional	R$100,00	Sábado
São Paulo	19:00	06:00	Leito	R$150,00	Diária
Porto União	13:00	18:00	Convencional	R$60,00	Diária
Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	[illegible]	[illegible]	Sábado
Blumenau	08:00	16:00	[illegible]	[illegible]	Diária
Itajaí	23:00	06:00	[illegible]	[illegible]	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	[illegible]	Terça
Urussanga	14:30	20:00	Leito	R$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R$80,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
Xanxerê	18:00	23:00	Executivo	R$60,00	Seg. a Sex.
Chapecó	13:00	21:00	Executivo	R$105,00	Seg. e Dom.
Xaxim	07:00	14:00	Executivo	R$165,00	Dom.
Válida para o período de 02/10/2008 a 30/11/2008.					

Figura 10.10 – Destacar linhas seletivamente.

Continue usando a tabela desenvolvida no `arquivo-10.4b.html` (Figura 10.4) para exemplificar esse efeito. As regras de estilo a acrescentar e o script para o efeito são mostrados a seguir.

▶ **CSS**:

```
.destacar-um {/* estiliza a célula que acaba de receber o ponteiro do mouse */
   background:#444;
   color:#fff;
   cursor:pointer;
   }
.tooltip {/* estiliza a caixinha da dica */
   width:180px;
   padding:2px 4px;
   margin-top:25px;
   border:1px solid #000;
   color:#333;
   background:#bfe2d8;
   position:absolute;
   }
```

▶ **jQuery**:

```
1.     <script type="text/javascript">
2.     $(document).ready(function() {
3.         $('table#horario tbody tr:odd').addClass('impar');
4.         $('table#horario tbody tr:even').addClass('par');
5.         $('table#horario tr#horizontal th').each(function(i) {
6.             var n = i + 1;
7.             var nomeColuna = $(this).text();
8.             $('table#horario tbody tr td:nth-child(' + n + ')')
9.                 .hover(function() {
10.                    $(this).addClass('destacar-um');
11.                    var textoCelula = $(this).text();
12.                    $('<div class="tooltip">Clique para destacar todas as linhas para<br />
13.                    <b> ' + nomeColuna + ': ' + textoCelula + '</b></div>').prependTo(this);
14.                    $(this).click(
15                         function() {
16                         $('table#horario tbody tr').removeClass('destacar');
17.                        $('.tooltip').remove();
18.                        $('table#horario tbody tr:contains(' + textoCelula + ')')
                               .addClass('destacar');
19.                    });
20.                },
21.                function() {
22.                    $(this).removeClass('destacar-um');
23.                    $('.tooltip').remove();
```

Linha(s)	Descrição (cont.)
Linhas 14 a 17	Tendo atingido a situação mostrada na figura 10.10, o usuário clica a célula e o script remove a classe destacar porventura existente em qualquer célula. Isto vai funcionar a partir do segundo clique para limpar um destaque anterior e acrescentar novo destaque. A seguir, o script remove o `div .tooltip`.
Linha 18	Seleciona todas as linhas da tabela que contenham o texto igual ao da célula clicada (mesmo dado) e destaca a linha. Veja o efeito na figura 10.11.
Linhas 21 a 24	Função para remover o destaque e o tooltip quando o usuário retira o ponteiro do mouse da célula sem clicá-la.

Na figura 10.11, mostramos o efeito final do destaque seletivo após o usuário ter clicado uma célula contendo a palavra "Executivo".

Livro jQuery do Maujor | Tabelas - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

Viação Alfa - Horários

Destino	Saída	Chegada	Classe	Tarifa	Freqüência
Brusque	06:45	14:30	Convencional	R$80,00	Diária
Joinville	12:30	21:00	Convencional	R$100,00	Sábado
São Paulo	19:00	06:00	Leito	R$150,00	Diária
Porto União	12:00	16:00	Convencional	R$60,00	Diária
Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	Leito	R$150,00	Sábado
Blumenau	08:00	16:00	Convencional	R$100,00	Diária.
Itajaí	23:00	06:00	Executivo	R$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R$80,00	Terça
Urussanga	14:30	20:00	Leito	R$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R$80,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
Xanxerê	18:00	23:00	Executivo	R$60,00	Seg. a Sex.
Chapecó	13:00	21:00	Executivo	R$105,00	Seg. e Dom.
Xaxim	07:00	14:00	Executivo	R$165,00	Dom.

Válida para o período de 02/10/2008 a 30/11/ 2008.

Passe o mouse sobre qualquer célula.

Figura 10.11 – Destaque seletivo, efeito final.

O script em questão apresenta um defeito que precisa ser corrigido. Caso o usuário clique uma célula contendo um determinado dado e este conste também de uma outra coluna que não a coluna clicada, a respectiva linha será destacada indevidamente. No caso em questão, isto fica bem evidente para as colunas que contêm os horários de saída e chegada dos ônibus. Um exemplo prático evidencia o defeito do script: considere que o usuário deseja destacar as linhas da tabela que contêm chegadas de ônibus às 23:00h. Clicando a célula da coluna Chegada e linha Caçador, cujo conteúdo é 23:00, deveriam ser destacadas as linhas Caçador e Xanxerê, que contêm chegadas de ônibus às 23:00h.

Na figura 10.12, veja o defeito do script nessa situação.

Livro jQuery do Maujor | Tabelas - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

Viação Alfa - Horários

Destino	Saída	Chegada	Classe	Tarifa	Frequência
Brusque	06:45	14:30	Convencional	R$80,00	Diária
Joinville	12:30	21:00	Convencional	R$100,00	Sábado
São Paulo	19:00	06:00	Leito	R$150,00	Diária
Porto União	13:00	18:00	Convencional	R$60,00	Diária
Caçador	00:30	23:00	Executivo	R$210,00	Seg. e Dom.
Sorocaba	21:15	23:45	Leito	R$150,00	Sábado
Blumenau	08:00	16:00	Convencional	R$100,00	Diária.
Itajaí	23:00	06:00	Executivo	R$165,00	Qua. e Sáb.
Lajes	16:45	19:30	Executivo	R$80,00	Terça
Urussanga	14:30	20:00	Leito	R$100,00	Sábado
Canoinhas	19:30	07:30	Leito	R$190,00	Seg. a Sex.
S. Fco. Sul	01:00	07:00	Convencional	R$80,00	Qua. a Sex.
Joaçaba	06:30	12:00	Executivo	R$70,00	Seg. e Qua.
Jaraguá do Sul	23:15	08:45	Leito	R$250,00	Sábado
Xanxerê	18:00	23:00	Executivo	R$60,00	Seg. a Sex.
Chapecó	13:00	21:00	Executivo	R$105,00	Seg. e Dom.
Xaxim	07:00	14:00	Executivo	R$165,00	Dom.

Válida para o período de 02/10/2008 a 30/11/ 2008.

Passe o mouse sobre qualquer célula da tabela

Figura 10.12 – Destaque seletivo, defeito do script.

Note que a linha para a cidade de Itajaí foi destacada indevidamente, pois o horário de chegada ali é 06:00h e não 23:00h como se queria. O script não está considerando somente a coluna cuja célula foi clicada para aplicar o efeito, ao contrário, está aplicando-o a qualquer célula da tabela que contenha o mesmo texto da célula clicada. Altere o script para corrigir isso.

Comece localizando a linha do script mostrado anteriormente, que adiciona a classe `destacar` às linhas da tabela. Dê uma olhada no script anterior e tente identificar qual é o número dessa linha.

Linha 18 é a resposta. Veja a seguir essa linha:

```
18.    $('table#horario tbody tr:contains(' + textoCelula + ')').addClass('destacar');
```

Aqui o script procura todas as linhas da tabela que contenham um texto igual ao texto da célula clicada e destaca a linha. Aqui o script é falho porque deveria procurar não por linhas que contenham o texto, mas por células da coluna clicada que contenham o texto para posteriormente selecionar as linhas de tais células e destacá-las. Escreva novamente a linha 18, com a correção proposta:

```
18.    $('table#horario tbody tr td:nth-child(' + n + '):contains(' + textoCelula + ')')
       .parents('tr').addClass('destacar');
```

Note que se criaram duas linhas de título para as cidades do norte e sul, linhas estas marcadas com célula de cabeçalho. O acréscimo na marcação da tabela, além das alterações introduzidas nas linhas de dados, é mostrado a seguir.

▶ **HTML**:

```
   ...
<tbody>
       <tr class="sub">
          <th colspan="6">Cidades no NORTE do Estado de Santa Catarina</th>
       </tr>
       ...linhas da tabela...
       <tr class="sub">
          <th colspan="6">Cidades no SUL do Estado de Santa Catarina</th>
       </tr>
       ...linhas da tabela...
<tbody>
...
```

Atribuiu-se uma classe para os novos cabeçalhos e acrescentaram-se as regras CSS mostradas a seguir:

```
tr.sub {/* estiliza os dois novos cabeçalhos com uma cor de fundo e de texto */
   background:#bfe2d8;
   color:#036;
   }
img.maismenos {/* para as imagens dos sinais de mais e menos nos cabeçalhos */
   border:none; /* retira a borda-padrão para imagens que são links */
   margin-right:10px; /* afasta a imagem do texto do cabeçalho */
   vertical-align:middle; /* centra a imagem verticalmente com o texto */
}
```

Note, na figura 10.13, que ao lado dos dois novos cabeçalhos há o sinal de menos (-) indicando o link para fechar. Esse sinal é uma imagem ali posicionada com o uso do script para não aparecer quando JavaScript estiver desabilitado no navegador.

Observe o script para o efeito proposto:

```
1.     <script type="text/javascript">
2.     $(document).ready(function() {
3.        var mais = '<a href="#">
4.        <img src="mais.gif" alt="Revelar/ocultar cidades" class="maismenos" />
5.        </a>'
6.        $('table#horario tbody tr:not(.sub):even').addClass('impar');
7.        $('table#horario tbody tr:not(.sub)').hide();
8.        $('.sub th').css({borderBottom:'1px solid #333'}).prepend(mais);
9.           $('img', $('.sub th'))
```

```
10.             .click(function(event) {
11.             event.preventDefault();
12.                 if (($(this).attr('src')) == 'menos.gif') {
13.                     $(this).attr('src', 'mais.gif')
14.                     .parents()
15.                     .siblings('tr').hide();
16.                 } else {
17.                         $(this).attr('src', 'menos.gif')
18.                     .parents().siblings('tr').show();
19.                 };
20.             });
21.     });
22. </script>
```

[arquivo-10.6a.html]

Código comentado:

Linha(s)	Descrição
Linhas 3 a 5	Cria uma variável com nome `mais` e armazena nela a marcação HTML para um link morto em uma imagem de um sinal de mais (+) contida no arquivo `mais.gif`. Coloca ainda o atributo `alt` na imagem e a classe `maismenos` para fins de estilização.
Linha 6	Adiciona a classe para obter o efeito zebra na tabela. Aqui convém notar que se usou a pseudoclasse `:not()`, para excluir do efeito zebra a linha dos dois novos cabeçalhos na tabela. Isto garante que a linha logo abaixo da linha de cabeçalho tenha sempre a mesma estilização (a mesma cor de fundo do efeito zebra).
Linha 7	Esconde todas as linhas da tabela que não sejam cabeçalhos. É erro comum esconder com o uso da declaração CSS `display:none` e, posteriormente, revelar com o uso de script. Não cometa esse erro, pois você tornará sua página inacessível caso o navegador não ofereça suporte para JavaScript. Se você pretende esconder conteúdos, comece com estes à mostra e esconda-os com o uso de script.
Linha 8	Adiciona uma borda inferior sólida à linha de cabeçalho e coloca na respectiva célula a imagem do sinal de mais (+) como um link.
Linha 9	Seleciona a imagem contida na célula de cabeçalho (inserida na linha 8).
Linha 10	Define que uma função deverá ser executada caso haja um clique da imagem do sinal de mais (+).
Linha 11	Impede que ao receber um clique o link morto seja seguido. Caso contrário, a página "pula" para o topo.

Outro aperfeiçoamento que se poderia introduzir no script é o fornecimento de um link contendo o texto "Abrir todas" que, ao ser clicado, revelaria toda a tabela ao mesmo tempo em que o texto mudaria para "Fechar todas" e que, ao ser clicado, obviamente, fecharia a tabela com o texto do link mudando para "Abrir todas". A esta altura, seu aprendizado com certeza já chegou a um ponto que o capacita a desenvolver esse script. Assim fica como sugestão o desenvolvimento do script para você testar seu conhecimento.

Advertência

Como você constatou nos exemplos para revelar e esconder conteúdos de uma tabela, utilizaram-se os métodos `hide()` e `show()` para os efeitos. Examinando o funcionamento dos arquivos de exemplos, talvez você seja tentado a melhorar o script com o uso de `slideUp()`, `slideDown()`, `fadeOut()`, `fadeIn()`, `animate()` ou mesmo `hide(velocidade)`, `show(velocidade)`.

Infelizmente, tais métodos não são interpretados de modo correto por todos os navegadores, quando se trata de tabelas, e usá-los, nestes casos, poderá causar efeitos estranhos e indesejados. Portanto, use somente os métodos `hide()` e `show()` para esse efeito quando se trata de aplicá-lo em tabelas. Observe, na figura 10.15, um desses efeitos estranhos referidos.

Figura 10.15 – slideDown() no Firefox.

Usar `slideDown()` e `slideUp()` causa uma revelação do conteúdo, como a mostrada na figura 10.5, quando visualizada no navegador Firefox. Para o navegador IE (Internet Explorer), o comportamento se passa como para `hide()` e `show()` normalmente, exceto se usar velocidade. Nesse caso, embora o comportamento continue como para `hide()` e `show()`, a troca da imagem do sinal respeita a velocidade.

Utilizar `fadeOut()` e `fadeIn()` com ou sem velocidade funciona no Firefox, mas falha no IE.

Enfim, faça algumas experiências e constate as diferenças para chegar à seguinte conclusão: use somente `hide()` e `show()` para esse efeito até que as inconsistências sejam resolvidas.

CAPÍTULO 11

Efeitos em formulários

Neste capítulo, serão abordados alguns efeitos da biblioteca jQuery para emprego em formulários. É necessário extremo cuidado ao se projetar scripts rodando no lado do cliente, como são os scripts jQuery, quando se trata de adicionar funcionalidades a formulários. Ao contrário da maioria dos conteúdos de uma página web, formulários destinam-se a possibilitar coleta de dados do usuário e envio deles ao servidor.

11.1 Validação de formulários

Dados entrados pelo usuário em campos de um formulário devem ser verificados antes de ser enviados ao servidor. A verificação se faz não somente para evitar entrada de código malicioso no servidor, via formulário, mas também para não permitir entrada de dados com formato diferente daquele para os quais o script de processamento no servidor foi desenvolvido. Por exemplo: se um campo deve receber um número, verifique se o usuário entrou um número nesse campo antes de enviá-lo ao servidor.

Assim, esteja consciente de que validar dados de formulários com o uso de JavaScript é uma técnica que não oferece segurança, apesar de ser adotada na maioria dos formulários existentes na web. Não abra mão de scripts rodando no lado do servidor para validar seus formulários. Feita a ressalva, veja alguns efeitos jQuery para formulários.

11.2 Placeholder para campos

Placeholder é um texto-padrão inserido em um campo de formulário fornecendo uma indicação genérica do formato de dado esperado no campo. Em geral, a fonte

do texto para o placeholder é estilizada em uma cor mais clara que a prevista para a fonte do texto entrado pelo usuário. Em um campo de formulário contendo um placeholder, tão logo o usuário entre no campo com o objetivo de preenchê-lo, o texto do placeholder é automaticamente retirado.

No primeiro efeito a estudar, você irá usar um campo de texto destinado à busca, ou simplesmente um campo de busca. O objetivo será retirar o rótulo do campo da sua posição e inseri-lo como placeholder.

A marcação HTML é mostrada a seguir:

```
<form action="" method="get" id="formulario">
   <label for="busca">Busca</label>
   <input name="busca" type="text" id="busca" />
</form>
```

Na figura 11.1, mostramos a renderização da página antes e depois da aplicação do efeito. Utilizou-se uma só tela para mostrar as duas situações, com a finalidade de simplificar.

Figura 11.1 – Campo de busca com rótulo como placeholder.

O rótulo do campo de busca é o texto inserido na marcação dentro do elemento `label`. Todo campo de formulário deve associar-se a um rótulo que tem por finalidade cumprir critérios de acessibilidade.

Observe o script para obter o efeito:

```
1.    <script type="text/javascript">
2.    $(document).ready(function() {
3.       var textoDefault = $('#formulario label').remove().text();
4.       $('#busca').val(textoDefault)  .css('color', '#999')
5.       .focus(function() {
6.          if ($(this).val() == textoDefault) {
7.             $(this).css('color', '').val('');
8.          }
9.       }).blur(function() {
10.         if ($(this).val() == ' ') {
```

```
11.             $(this).val(textoDefault).css('color', '#999');
12.         }
13.     });
14. });
15. </script>
```

 [arquivo-11.2a.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Armazenou-se em uma variável o rótulo do campo e, com o uso do método `remove()`, retirou-se o rótulo da marcação. É importante notar que o método `remove()` retira o elemento do DOM, mas permite acessar seus conteúdos normalmente. Ao contrário, o método `empty()` retira os conteúdos do elemento e conserva o elemento no DOM. Este é um conceito importante que diferencia um método do outro e se você não estiver atento à diferença, poderá empregar um ou outro de forma equivocada e não entender o porquê de seu script não funcionar. Caso utilizasse `empty()`, o script falharia.
Linha 4	Acessa o campo de busca e a este atribui um valor igual ao texto do rótulo, armazenado anteriormente na variável `textoDefault`. A seguir, cria uma declaração CSS para o texto, definindo para ele uma cor cinza-claro. O método `val()` manipula o atributo HTML `value`. Declarar `val('valor')` equivale a escrever na marcação `value="valor"`.
Linha 5	Define-se uma função a ser executada quando o campo de busca recebe o foco.
Linha 6	Testa o valor do campo de busca contra o `textoDefault`. No carregamento da página, o teste resulta verdadeiro, pois ao campo de busca se atribuiu o `textoDefault` conforme definido na linha 4.
Linha 7	Tendo recebido o foco e a condição sendo verdadeira, remove-se a regra CSS para o texto no campo, bem como o próprio texto, deixando o campo limpo para receber a entrada de dado do usuário.
Linha 9	Aqui, define-se uma função a ser executada quando o foco é retirado do campo.
Linha 10	Testa para saber o que aconteceu com o campo após ter recebido o foco. Duas são as possibilidades: o usuário preencheu e saiu ou deixou em branco e saiu.
Linha 11	Se deixou em branco e saiu, repõe o `textoDefault` no campo. Caso contrário, deixa o campo com o dado preenchido para posterior validação.

Outra solução comum nesses casos consiste em usar como placeholder um texto que não o rótulo do campo, mantendo este no seu local original, tal como mostra a figura 11.2.

Figura 11.2 – Campo de busca com título como placeholder.

Note que o rótulo permanece em seu lugar e o texto "palavra chave" foi usado como placeholder.

Entre as várias soluções possíveis, selecionaram-se duas apresentadas a seguir.

Primeira solução

Inserir o atributo `title` no elemento `label` da marcação e usar o valor desse atributo como placeholder. Observe o acréscimo na marcação HTML e a alteração na linha 3 do script anterior.

- HTML:

```
<label for="busca" title="palavra chave">Busca</label>
```

- jQuery:

```
...
var textoDefault = $('#formulario label').attr('title');
...
```

[arquivo-11.2b.html]

Código comentado:

Linha	Descrição
Linha 3	Nesta solução, utilizou-se o método `attr('title')` para acessar o valor do atributo `title` do elemento `label` e atribuí-lo à variável `textoDefault`. Todo o restante do script permanece como na solução anterior.

Segunda solução

A segunda solução consiste em não introduzir alteração na marcação, criando o texto do placeholder via script. Nesse caso, bastaria alterar a linha 3 conforme mostrado a seguir.

```
var textoDefault = 'palavra chave';
```

```
4.           .css({display:'none',
5.              border:'1px solid #add6ef',
6.              padding:'2px 4px',
7.              background:'#d6e2e5',
8.              marginLeft:'10px'
9.           });
10.          $('.dica').focus(function() {
11.             $(this).next().fadeIn(1500);
12.          }).blur(function() {
13.             $(this).next().fadeOut(1500);
14.          });
15.       });
16.    </script>
```

[arquivo-11.3a.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Utilizou-se o seletor do tipo `E + F` para selecionar os elementos `span` imediatamente após o elemento com a classe `dica`. Veja na marcação que são os elementos `span` que contêm o texto da dica. Seletores do tipo `E + F` casam com o elemento `F` que vem imediatamente após um elemento `E`.
Linhas 4 a 9	Regras CSS destinadas a estilizar a caixa contendo a dica e a escondê-la quando a página for carregada. Lembre-se de que um impulso inicial pode levar o desenvolvedor a definir essas regras de estilo na folha de estilos da página. Você pode adotar essa solução sem problemas, desde que conserve a declaração `display:none` no script, não a transferindo para as CSS da página, pois assim fazendo, você bloqueia o acesso às dicas quando JavaScript não estiver disponível no navegador.
Linha 10	Define uma função a ser executada quando o campo recebe o foco.
Linha 11	Revela a dica que estava oculta usando o efeito `fadeIn()`.
Linha 12	Define uma função a ser executada quando o campo perde o foco.
Linha 13	Esconde a dica usando o efeito `fadeOut()`.

11.4 Desabilitar campos

Para exemplificar o efeito de desabilitar campos de um formulário, tome como base um formulário composto de três áreas distintas, cada uma delas destinadas a um fabricante de automóvel, selecionável por um campo de formulário do tipo botão radio (`radio`). Para cada fabricante é apresentada uma série de campos de

formulário do tipo caixa de seleção (checkbox) que permite ao usuário escolher vários modelos de automóvel do fabricante selecionado. A marcação HTML (omitiram-se alguns checkboxes para reduzir o tamanho do código) é mostrada a seguir:

```
<form action="" method="get" id="formulario">
<fieldset>
   <legend>Selecione um fabricante e escolha os modelos de sua preferência.</legend>
   <div class="mudar">
      <label class="fabricante"><input type="radio" name="fabricante" />
         Mercedes Benz</label>
      <label><input type="checkbox" />Classe A - Hacthback</label>
      <label><input type="checkbox" />Classe B - MPV</label>
      ...mais checkbox...
   </div>
   <div class="mudar">
      <label class="fabricante"><input type="radio" name="fabricante" />Ferrari</label>
      <label><input type="checkbox" />599 GTB</label>
      <label><input type="checkbox" />California 2009</label>
      ...mais checkbox...
   </div>
   <div class="mudar">
      <label class="fabricante"><input type="radio" name="fabricante" />Volvo</label>
      <label><input type="checkbox" />S40</label>
      <label><input type="checkbox" />S60</label>
      ...mais checkbox...
   </div>
</fieldset>
</form>
```

Note que rótulos (label) implícitos foram declarados para os campos do formulário. Isto possibilitou que ao se declarar float:left para os elementos label, estes carregassem consigo seus elementos input respectivos, facilitando o posicionamento e alinhamento dos campos do formulário. As demais regras CSS são triviais.

Diz-se que um rótulo de formulário é implícito quando o elemento label, além de conter o texto descritivo do campo, é também seu container. Rótulos não implícitos contêm somente o texto descritivo e devem ser marcados com o atributo for com nome igual ao nome do atributo id do campo.

Com a finalidade de posicionar e melhorar a apresentação do formulário em questão, criou-se a folha de estilos mostrada a seguir:

```
body {width:600px; font:80%/1.2 Arial, Helvetica, sans-serif; margin:30px auto;
   color:#666; padding:0;}
label {width:190px; display:block;float:left;}
```

```
fieldset {border:1px solid #ccc; padding:10px;}
legend {font-weight:bold; border:1px solid #ccc; padding:1px 4px;}
label.fabricante {font-weight:bold; margin:0;}
form div {margin:15px 0 0 0;}
.mudar, .fabricante {width:100%;overflow:auto;}
.mudar {background:#d6e2e5;}
.fabricante {background:#add6ef;padding:10px 0;}
```

[arquivo-11.4a.html]

Observe, na figura 11.4, a renderização do formulário proposto.

Figura 11.4 – Formulário básico proposto.

O formulário foi projetado para permitir a seleção de um fabricante apenas e vários modelos. Na figura 11.4, você pode notar que houve seleção de um fabricante e de modelos dos três fabricantes.

Não serão tecidas considerações sobre técnicas ou validações para evitar uma seleção ilegal como a mostrada, pois o objetivo é estudar a técnica de desabilitar campos de formulário. Contudo, vale lembrar que esta é uma solução que não deve estar isolada, pois caso JavaScript esteja indisponível no navegador, o script falhará. É recomendável que seja usada como alternativa simplificada para outro mecanismo, para desabilitar os campos.

Observe o script para obter o efeito de desabilitar os campos do formulário para as opções de fabricante não escolhidas:

```
1.    <script type="text/javascript">
2.        $(document).ready(function() {
3.            $('#formulario input:checkbox').attr('disabled', 'disabled');
4.            $('#formulario input:radio').click(function() {
5.                $('#formulario input:checkbox').removeAttr('checked')
                    .removeAttr('disabled');
6.                $(this).parents('.mudar').siblings('.mudar')
7.                    .find(':checkbox')
8.                    .attr('disabled', 'disabled');
9.            });
10.       });
11.   </script>
```

 [arquivo-11.4b.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Inicia-se desabilitando todos os checkboxes, para evitar que o usuário escolha os modelos sem ter escolhido um fabricante. Note o uso da pseudoclasse :`checkbox` no seletor.
Linha 4	Define-se uma função a ser executada quando o usuário clica um dos botões radio para a escolha do fabricante. Note o uso da pseudoclasse :`radio` no seletor.
Linha 5	Esta linha contém uma ação a ser executada somente no caso de o usuário ter escolhido um fabricante, selecionado os modelos e resolvido alterar a escolha do fabricante. Por que alguém faria isso? Realmente, não faz muito sentido, mas ao projetar formulários, procure simular qualquer situação de preenchimento e interação do usuário, por mais absurda que pareça. Ocorrendo tal situação, o script precisa limpar as seleções feitas antes de desabilitar as escolhas. É essa a ação contida nesta linha: seleciona todos os `checkboxes`, limpa os selecionados e habilita-os para nova escolha.
Linha 6	Encontra todos os elementos `div.mudar`, que são o elemento-pai dos conteúdos dos blocos de fabricantes que (no nosso caso são dois) não sejam o fabricante selecionado pelo usuário.
Linha 7	Encontra todos os checkboxes que não sejam os do fabricante selecionado pelo usuário.
Linha 8	Desabilita os checkboxes encontrados.

Observe, na figura 11.5, o efeito final com a escolha de um fabricante, alguns modelos e os modelos dos demais fabricantes desabilitados. Note que a escolha de um fabricante desabilita a escolha dos modelos dos demais, mas não desabilita a escolha do fabricante, possibilitando ao usuário mudar de opção.

Figura 11.5 – Formulário com efeito desabilitar.

Agora não é mais possível uma seleção inválida como aquela mostrada na figura 11.4.

11.5 Revelar campos

Para esta solução, a proposta é apresentar, inicialmente, ao usuário somente a opção de escolha do fabricante. Feita a opção, os checkboxes para a escolha dos modelos serão revelados.

A seguir, o script para obter o efeito proposto:

```
1.    <script type="text/javascript">
2.        $(document).ready(function() {
3.            $(':checkbox').parents('label').hide();
4.            $(':radio').wrap('<a href="#"></a>');
5.            $(':radio').parents('label').click(function() {
6.                $(this).siblings().fadeIn(1500);
7.                $(this).parent().siblings('.mudar')
8.                    .children('label:not(.fabricante)')
9.                    .slideUp().find(':checked').each(function() {
10.                       $(this).removeAttr('checked');
11.                   });
12.           });
13.       });
14.   </script>
```

[arquivo-11.5a.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Encontra todos os elementos `label` que sejam elementos-pai de `checkbox` e esconde-os. Por se ter usado rótulo implícito, não só os rótulos, mas também os elementos `input` a estes associados serão escondidos.
Linha 4	Colocou-se como container para os elementos radio um link morto. Você saberia explicar a razão desse link? Se pensou em navegação por teclado, acertou. Sem um link, o usuário navegando com auxílio do teclado não conseguiria abrir as opções de modelos de um fabricante.
Linha 5	Definiu-se uma função a ser executada quando o usuário escolhe um fabricante. Outra vez o uso de rótulo implícito facilita o script. Clicar o botão de escolha ou o nome do fabricante, indiferentemente, dispara a execução da função.
Linha 6	Seleciona os elementos-irmão do elemento `label` clicado e revela-os.
Linha 7	Encontra o elemento-pai do elemento `label` clicado. É um `div` com a classe `mudar`, a seguir encontra seus elementos-irmão com a classe `mudar`. São os outros dois divs.
Linha 8	Encontra os elementos-filho das dois divs que não sejam os elementos `radio` de escolha dos fabricantes. São os checkboxes para a escolha dos modelos.
Linha 9	Esconde o encontrado na linha anterior e procura por aqueles que tenham sido marcados pelo usuário.
Linha 10	Faz um loop pelos marcados e desmarca-os. Este procedimento é necessário nos casos em que o usuário faz uma escolha de modelos e resolve mudar o fabricante. É preciso desmarcar os escolhidos antes de fechar o que estava aberto para, em seguida, revelar a nova escolha.

Na figura 11.6, constam a apresentação inicial do formulário e a apresentação após ter sido feita a escolha de um fabricante.

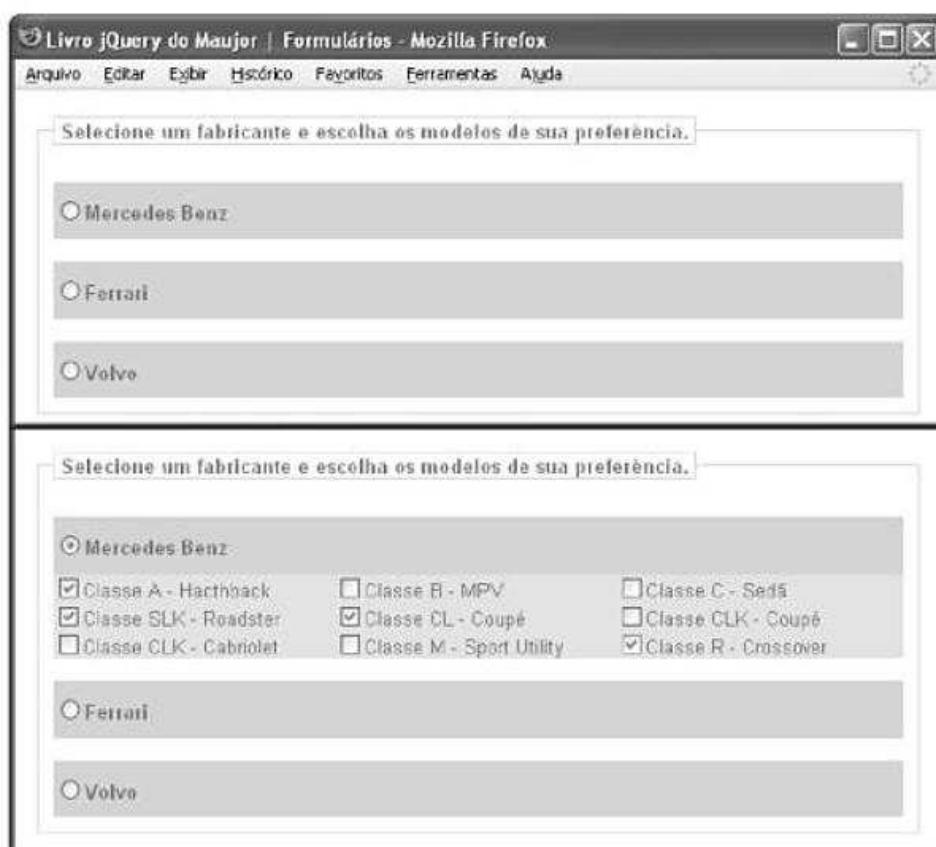



Figura 11.6 – Formulário revelando campos.

11.6 Elemento legend

Veja, nesta seção, uma técnica que não é exclusiva de formulários e pode ser aplicada a outros cenários.

Sabe-se que os navegadores, para alguns elementos HTML, apresentam restrições e inconsistências na renderização quando servidos com determinadas regras CSS. Os elementos `caption` para tabelas e `legend` para formulários se enquadram nessa categoria. Outros como `select` e `option` simplesmente ignoram certas regras CSS.

Nesses casos, a opção é apresentar tais elementos com estilização mínima e conseguir efeitos de apresentação mais sofisticados com o uso de script. A técnica, para os elementos `legend` e `caption` cujos conteúdos são textuais, consiste em substituir o elemento por um outro que aceite regras CSS sem restrições.

Utilize neste exemplo o `arquivo-11.5a.html` desenvolvido na seção anterior. Substitua o elemento `legend` por um elemento `h4` (cabeçalho nível quatro) ao qual se atribuirá a classe `legenda`. Altere também a borda do elemento `fieldset` de sólida para pontilhada. As regras CSS a acrescentar são as seguintes:

11.7 Selecionar todos

Um cenário comum em formulários é aquele no qual se apresenta ao usuário uma série de campos de seleção, sendo permitida a seleção de mais de uma opção, tal como mostrado na seleção dos modelos de automóveis nos exemplos anteriores.

É de boa técnica fornecer ao usuário um mecanismo para selecionar e desmarcar todas as opções de uma só vez. Para demonstrar o desenvolvimento e funcionamento do script que implementa essa funcionalidade, utilize o formulário cuja marcação HTML é mostrada a seguir:

```
<form action="" method="get" id="formulario">
<fieldset>
<legend>Escolha os modelos de sua preferência.</legend>
<div class="mudar">
   <label><input type="checkbox" />Classe A - Hacthback</label>
   <label><input type="checkbox" />Classe B - MPV</label>
   ...mais checkboxes...
   <label><input type="checkbox" />P4/5</label>
</div>
</fieldset>
</form>
```

A renderização do formulário é mostrada na parte superior da figura 11.8. Na parte inferior, consta o formulário após ter sido desenvolvido e incorporado na página, o script jQuery proposto nesta seção.

Figura 11.8 – Formulário, selecionar todos.

O script para a funcionalidade de selecionar todos é mostrado a seguir:

```
1.    $(document).ready(function() {
2.       var selecionaTodos = $('<label><input type="checkbox" class="todos" /><b>
            Selecionar todos</b></label>');
3.       $(selecionaTodos).insertBefore('.mudar')
4.       var todosCheckboxes = $('.mudar').find(':checkbox');
5.       $('.todos').click(function() {
6.          if (this.checked) {
7.             $(this).next().text('Desmarcar todos')
8.             $(todosCheckboxes).attr('checked', 'checked');
9.          } else {
10.            $(this).next().text('Selecionar todos')
11.            $(todosCheckboxes).removeAttr('checked');
12.         }
13.      });
14.      $('<h4 class="legenda"></h4>').insertAfter('legend');
15.      ...
16.   });
17.   </script>
```

 [arquivo-11.7a.html]

Código comentado:

Linha(s)	Descrição
Linha 2	Cria uma variável e armazena a marcação HTML para o `checkbox` a ser inserido na página com a finalidade de fornecer a opção de seleção de todos os modelos. Como esta funcionalidade estará disponível apenas em navegadores com JavaScript habilitado, não se deve colocar o `checkbox` diretamente na marcação, e sim inseri-lo via script.
Linha 3	Insere o `checkbox` criado imediatamente antes do `div` container para as opções de modelos. Utilizou-se `insertBefore()` e não `prependTo()` porque neste caso a inserção seria dentro do `div` e o `checkbox` estaria sujeito às regras de estilo para os `checkboxes` de escolha dos modelos. Assim, iria se posicionar ao lado da primeira opção de escolha e não acima de todas elas como se desejava. Faça uma cópia do arquivo, altere e comprove o comportamento com `prependTo()`.
Linha 4	Armazena em uma variável todos os checkboxes para a escolha dos modelos.
Linha 5	Define uma função a ser executada quando o usuário seleciona o `checkbox` que foi inserido para selecionar todas.
Linha 6	Verifica se a seleção de todos os modelos está marcada. Logo após o primeiro clique do usuário, esta condição é verdadeira e o script segue para a linha 7.

Note que se acrescentou à marcação HTML da página um botão de envio do formulário ao qual se atribuiu um `id="enviar"` e que, quando clicado, fará o script de validação ser acionado. Não há inserção de JavaScript na marcação do botão para acionar a execução da função de validação.

O script para a funcionalidade de validação é mostrado a seguir:

```
1.     $(document).ready(function() {
2.        $('#enviar').click(function() {
3.           var i = $('.mudar input:checked').size();
4.           if (i == 0 || i > 5) {
5.              alert('Por favor, selecione no mínimo um e no máximo cinco modelos');
6.           }
7.        });
8.     });
9.     </script>
```

 [arquivo-11.7a.html]

Código comentado:

Linha(s)	Descrição
Linha 2	Define a função de validação a ser executada quando o usuário clica o botão "Enviar".
Linha 3	Usa o método `size()` para armazenar a quantidade de modelos de automóvel selecionados pelo usuário.
Linha 4	Testa a quantidade selecionada e insere na tela a caixa de alerta JavaScript se a condição for verdadeira.

CAPÍTULO 12
Imagens

Neste capítulo, serão estudadas as técnicas de manipulação de imagens com o uso da biblioteca jQuery. Serão abordados alguns efeitos simples destinados a ampliar imagens e também a construção de galerias de imagens. O objetivo principal será ampliar o conhecimento das técnicas gerais de criação jQuery. Existem vários plug-ins para manipulação de imagens e desenvolvimento de galerias e slide-show de imagens. Ao término deste capítulo, você, além de atingir o objetivo proposto, estará em condições de desenvolver scripts simples para ampliação de imagens e construção de galerias de imagens com recursos razoáveis. Havendo necessidade de inserir em seu projeto uma galeria de imagens com recursos avançados, consulte o repositório de plug-ins em `http://plugins.jquery.com/`.

12.1 Introdução

Inserir imagens em uma página web é um recurso empregado na maioria dos sites. Imagem em páginas web, além de transmitir informação, torna a leitura da página mais agradável. A internet foi inventada por Tim Berners-Lee com o propósito inicial de ser um meio de troca de informações entre cientistas. Os documentos de natureza científica, em geral, são criados com o uso de texto puro, sem necessidade de elaborados recursos de apresentação. Na proposta inicial de Tim, não estava prevista a inserção de imagens nem de nenhum tipo de mídia ou objeto nos documentos a transitar pela internet.

A internet tornou-se uma rede popular e difundiu-se muito, graças, em grande parte, à invenção dos elementos HTML para inserção de mídia, a começar pelo elemento para inserir imagens, com o consequente aparecimento do primeiro navegador gráfico, o Mosaic. A invenção do Mosaic marcou o início do desenvolvimento da internet.

12.1.1 Imagens acessíveis

Quanto à sua finalidade, as imagens em um documento podem ser classificadas em dois grupos, conforme descritos a seguir.

12.1.1.1 Imagens decorativas

Como o próprio nome sugere, são aquelas destinadas a melhorar o efeito visual da página, não tendo nenhuma influência ou relação com a transmissão da informação. Se forem retiradas da página, não causarão outro impacto que não o de tornar mais limitada a apresentação visual. Estas não merecem atenção sob o ponto de vista da acessibilidade e devem ser marcadas com o uso do atributo `alt` vazio, conforme mostrado a seguir:

```
<img src="cd.jpg" alt="" />
```

12.1.1.2 Imagens para transmitir informação

São imagens criadas com a finalidade de complementar, esclarecer ou substituir um conteúdo textual ou um conteúdo que seja indispensável para o entendimento da informação. O exemplo clássico de uma imagem dessa natureza é aquela criada para substituir os cabeçalhos, em um documento, que devam ser apresentados com um tipo de fonte decorativa não disponível na máquina dos usuários.

As Recomendações do W3C para acessibilidade preconizam que nesses casos se deve fornecer uma alternativa textual para a imagem, com a finalidade de não bloquear acesso à informação. Lembre-se de que usuários que estejam navegando com um leitor de tela ou com imagens desabilitadas não tomarão conhecimento do conteúdo da imagem a menos que se forneça tal alternativa.

Existem dois métodos básicos para se fornecer o equivalente textual para uma imagem que transmita informação ou que seja indispensável para o entendimento da informação.

Atributo `alt`

Para imagens cujo equivalente textual seja um texto curto, sem necessidade de descrição detalhada, coloca-se o texto como valor do atributo `alt`, conforme o exemplo mostrado a seguir:

```
<img src="cd.jpg" alt="Capa do CD dos Beatles" />
```

Atributo `longdesc`

Para imagens que requerem como equivalente textual um texto explicativo longo, usa-se o atributo `longdesc` (long description, que significa descrição longa), conforme o exemplo mostrado a seguir:

```
<img src="cd.jpg" alt="Capa do CD dos Beatles" longdesc="cd_beatles.html" />
```

O atributo `longdesc` aponta para um arquivo de texto onde se encontra a descrição detalhada da imagem e é um suplemento ao atributo `alt`, devendo ser usado com este.

12.2 Ampliação de imagens

Agora você irá examinar algumas técnicas de ampliação. É comum em páginas web, com a finalidade de não poluir visualmente a página com inserção de imagens grandes, fornecer uma versão miniatura da imagem, conhecida como thumbnail, que, ao ser clicada, fornece a visualização da imagem em seu tamanho natural.

Em tais situações, o thumbnail é um link apontando para a versão ampliada da imagem. Sempre que se marcar uma imagem como link, os navegadores, por padrão, inserem uma borda, normalmente na cor azul típica da indicação de links, ao redor da imagem. A primeira providência a se tomar é retirar essa indesejável borda, declarando a seguinte regra CSS:

```
img {border:none;}
```

Ou usar um outro seletor mais específico que tenha como alvo somente as imagens que são link, como mostrado a seguir:

```
a img {border:none;}
```

Considere a marcação HTML a seguir:

```
...
<div id="galeria">
<a href="1.jpg"><img src="1.jpg" alt="Descrição da foto" width="120" height="90"
   class="foto" /></a>
</div>
<span>Clique para ampliar</span>
...
```

Trata-se de um thumbnail inserido como link em uma página, tendo logo abaixo um texto com uma dica instruindo o usuário a clicá-lo para visualizar a imagem ampliada. A imagem encontra-se no arquivo `1.jpg`, no seu tamanho natural, e será utilizada para obter o thumbnail forçando a redução de suas dimensões com o uso dos atributos `width` e `height` no elemento de marcação `img`. Assim sendo, não será necessário gerar duas imagens, uma para o thumbnail e outra em seu tamanho natural. Considere que não será preciso gerar um thumbnail personalizado, como aqueles constituídos por uma porção da imagem ampliada.

Note que se utilizou como container para a imagem um `div` com sua respectiva `id` denominada `galeria`. Esse container é perfeitamente dispensável, mas foi ali inserido tendo em vista as próximas etapas deste estudo nas quais se estenderá essa marcação e o container será indispensável.

Observe, na figura 12.1, a renderização da marcação proposta e o resultado quando o usuário clica o thumbnail.

Figura 12.1 – Ampliação de imagem em página web sem script.

Ao clicar a imagem, o usuário obtém a visualização da imagem em seu tamanho natural. Note que a imagem ampliada foi apresentada na mesma janela onde se encontrava o thumbnail, pois não se usou o atributo `target` no elemento link definindo abertura em uma nova janela.

A propósito, documentos que utilizem uma DTD Strict não são válidos se usarem o atributo `target` para links. Em documentos que admitem o uso desse atributo, sempre que um link abrir outra janela, informe, junto ao link, esse comportamento.

Primeira etapa

Em uma primeira etapa, você irá estudar um script simples para obter a ampliação da imagem. Complementarmente, com a finalidade de mostrar uma técnica de manipulação de textos existentes em páginas, será alterada a maneira de apresentação da dica para clicar a imagem.

Na figura 12.2 mostramos o efeito final com a dica posicionada sobre a imagem.

Figura 12.2 – Ampliação de imagem em página web com o uso de script.

Observe as regras de estilo e o script para obter o efeito.

▶ **CSS**:

```
<style type="text/css" media="all">.foto {border:none;}
#galeria {width:500px; position:relative;}
.abre-fecha {/* esta classe foi inserida pelo script no elemento span que marca a
   dica ao usuário */
   position:absolute;
   background:#add6ef;
   left:0;
   top:0;
   text-align:center;
   font-size:11px;
   cursor:pointer;
   }
</style>
```

▶ **jQuery**:

```
1.    <script type="text/javascript">
2.    $(document).ready(function() {
3.        var af = $('#galeria + span').addClass('abre-fecha')
4.        .css({opacity:0.6, width:120});
5.        $('#galeria').prepend(af)
6.            .click(function(event) {
7.                event.preventDefault();
8.                if ($('.foto').attr('width') > 120) {
9.                    $('.foto').attr({width:120, height:90});
10.                   $('.abre-fecha').text('Clique para ampliar')
11.                       .css({width:$('.foto').attr('width')});
12.               } else {
13.                   $('.foto').attr({width:500, height:375});
14.                   $('.abre-fecha').text('Clique para fechar')
15.                       .css({width:$('.foto').attr('width')});
16.               };
17.           });
18.       });
19.   </script>
```

[arquivo-12.2a.html]

Código comentado:

Linha(s)	Descrição
Linhas 3 e 4	Armazenou-se em uma variável um objeto jQuery que é descrito assim: seleciona-se o elemento span que se segue ao div#galeria (é o elemento da marcação que contém a dica ao usuário - Clique para ampliar - e a ele se atribui uma classe denominada abre-fecha) (veja regras de estilo) e, a seguir, define-se, para a dica, uma opacidade e uma largura igual à largura do thumbnail.

Linha(s)	Descrição (cont.)
Linha 5	Insere-se dentro e no início – `prepend()` – do `div#galeria` a dica, conforme modificada anteriormente.
Linha 6	Define-se uma função para ser executada quando o usuário clica o `div#galeria`.
Linha 7	Impede que o clique na imagem cause a abertura de uma janela com a imagem ampliada. Equivale a `return false`.
Linha 8	Testa o tamanho da foto na tela, verificando se sua largura é maior que a do thumbnail (120px, no caso do exemplo em questão). Quando a página for carregada, a condição será falsa, pois nessa ocasião se apresenta o thumbnail. O script vai para a linha 12. Suponha que o usuário já ampliou a imagem e clicou-a para reverter ao estado inicial, mostrando novamente o thumbnail. Nesta situação, a condição de teste é verdadeira e você deve seguir com a linha 9. Caso se confunda no acompanhamento do funcionamento do script, leia a partir da linha 13 e depois volte para a linha 9.
Linha 9	Como a imagem apresentada é a ampliada, insere-se o atributo `width` para restabelecer a largura do thumbnail.
Linha 10	Altera-se o texto da dica.
Linha 11	Define-se o texto da dica com largura igual à do thumbnail.
Linha 13	Aqui começa o funcionamento do script quando o usuário clica o thumbnail para ampliá-lo. Insere os atributos definindo a largura e a altura da imagem ampliada.
Linha 14	Altera o texto da dica.
Linha 15	Define o elemento `span`, que contém a dica, com uma largura igual à da imagem ampliada.

Segunda etapa

Como você deve ter observado, o script desenvolvido na etapa anterior causa a ampliação e fechamento da imagem de forma abrupta. Nesta segunda etapa, você irá usar o método `animate()` para melhorar essas transições. Observe os acréscimos introduzidos no script anterior:

```
1.    <script type="text/javascript">
2.    $(document).ready(function() {
3.       var af = $('#galeria + span').addClass('abre-fecha')
4.       .css({opacity:0.6, width:120});
5.       $('#galeria').prepend(af)
6.       .click(function(event) {
7.          event.preventDefault();
8.          $('.abre-fecha').hide();
9.          if ($('.foto').attr('width') > 120) {
```

Código comentado:

Linha(s)	Descrição
Linhas 11 e 20	Acrescenta-se a mudança de opacidade para 20% durante a animação para ampliar e reduzir. Isto significa que ao final de cada animação, a imagem estará com sua opacidade reduzida para esse valor.
Linhas 12 e 21	Ao final da animação de ampliação e redução, executa-se uma função que restabelece a opacidade da imagem com velocidade igual a 1 segundo.

Quarta etapa

Considere que a animação obtida na etapa anterior satisfaz seus propósitos e, nesta etapa, você focará outro aspecto da ampliação da imagem. Note que se utilizou uma imagem gravada com as dimensões de 500 x 375px e, a partir dela, construiu-se um thumbnail com o uso dos atributos `width` e `height` no elemento `img` constante da marcação. Isto tirou flexibilidade do script, pois se em outro local do site você precisar usar um thumbnail e/ou imagem ampliada com outras dimensões, terá de reescrever a marcação e o script.

Nesta solução, torne o script reutilizável, possibilitando a personalização das dimensões das imagens. Note que em navegadores sem suporte para JavaScript, não há escolha para variar as dimensões a não ser alterar a marcação e/ou as dimensões da imagem original.

No seu script, crie variáveis para armazenar as dimensões. Com isso, basta mudar o valor das variáveis de acordo com a necessidade. As alterações no script anterior são mostradas a seguir:

```
1.    <script type="text/javascript">
2.    $(document).ready(function() {
3.       var larguraFoto = 800; var alturaFoto = 600;
4.       var larguraMini = 200; var alturaMini = 150;
5.       $('.foto').attr({width:larguraMini, height:alturaMini});
         var af = $('#galeria + span').addClass('abre-fecha')
         .css({opacity:0.6, width:larguraMini});
5.       $('#galeria').prepend(af)
6.          .click(function(event) {
7.             event.preventDefault();
8.             $('.abre-fecha').hide();
9.             if ($('.foto').attr('width') > larguraMini) {
10.               $('.foto').animate(
11.                  {width:larguraMini, height:alturaMini, opacity:0.2},
                     1500, function() {
```

```
12.                         $('.foto').animate ({opacity:1}, 1000);
13.                         $('.abre-fecha').show().text('Clique para ampliar');
14.                         .css({width:$('.foto').attr('width')});
15.                     });
16.                 $('.abre-fecha').text('Clique para ampliar')
17.                 .css({width:$('.foto').attr('width')});
18.             } else {
19.                 $('.foto').animate(
20.                     {width:larguraFoto, height:alturaFoto, opacity:0.2}, 1500,
                        function() {
21.                     $('.foto').animate ({opacity:1}, 1000);
22.                     $('.abre-fecha').show().text('Clique para fechar')
23.                     .css({width:$('.foto').attr('width')});
24.                 });
25.             };
26.         });
27.     });
28. </script>
```

[arquivo-12.2d.html]

No arquivo que ilustra essa solução, utilizaram-se a mesma imagem e marcação HTML das soluções anteriores e definiram-se tanto o thumbnail como a imagem ampliada com dimensões diferentes das adotadas nessas soluções. Faça uma cópia do arquivo que ilustra este exemplo e altere, no script, as dimensões do thumbnail e da imagem ampliada, para verificar na prática o funcionamento do script.

Quinta etapa

Para esta etapa, você irá estudar uma solução de ampliação de imagem que simula o efeito lightbox. Esse efeito consiste em posicionar a imagem ampliada no centro da tela sobre uma máscara na cor preta transparente ocupando toda a tela.

Para esta etapa, utilize uma marcação HTML e CSS conforme os códigos mostrados a seguir.

▶ **CSS**:

```
1.  <style type="text/css" media="all">
2.      body {margin:20px 0; padding:0; background:#d6e2e5;}
3.      #tudo {width:600px; font:80%/1.2 Arial, Helvetica, sans-serif; margin:0 auto;
            padding:0 20px; color:#666; background:#fff; border:1px dotted #999;}
4.      .foto{border:none; margin:10px;}
5.      .left {float:left;}
6.      .right {float:right;}
```

Linha(s)	Descrição (cont.)
Linha 14	Acrescenta o caminho para a imagem ampliada, no elemento `img` criado anteriormente.
Linhas 15 a 17	Define regras CSS de posicionamento para a imagem ampliada. Note que a folha de estilo da página definiu posicionamento absoluto (em relação ao elemento `body`) para a imagem. As coordenadas `left` e `top` aqui definidas posicionam a imagem no centro da tela. Veja figura 12.4 para detalhe.
Linha 18	Coloca, no documento, o elemento `img` criado anteriormente com seu conteúdo e regras CSS. A seguir, define uma função a ser executada quando o usuário clica a imagem ampliada.
Linha 19	Faz desaparecer a imagem ampliada com o uso do efeito `fadeOut()` e velocidade de 1 segundo.
Linha 20	Faz desaparecer a máscara com o uso do efeito `fadeOut()` na velocidade de 1,5 segundo.

Na figura 12.4, mostramos o efeito final e as dimensões citadas nas linhas 16 e 17 do script para esclarecer o posicionamento no centro da tela.

Figura 12.4 – Efeito lightbox final.

Linha(s)	Descrição (cont.)
Linha 4	Define uma função a ser executada quando o usuário clica uma imagem da galeria.
Linha 5	Impede que o navegador siga o link para o arquivo da imagem. Afinal, você é que irá definir como a imagem será ampliada, não desejando simplesmente um link para o arquivo da imagem.
Linha 6	Esconde toda a galeria.
Linha 7	Testa para saber se o clique foi em um thumbnail ou em uma imagem ampliada. A condição de teste é para saber se a largura da imagem é maior que a do thumbnail (a largura é de 100px no exemplo). Neste caso, a imagem clicada foi a ampliada. Quando a página é carregada, o primeiro clique será obrigatoriamente em um thumbnail e o teste resulta falso com o script indo para a linha 11. Supondo que uma imagem ampliada foi aberta e o usuário realiza um clique para fechá-la, o script segue para a linha 8.
Linha 8	Reduz as dimensões da imagem ampliada para as dimensões do thumbnail, para recolocá-la na galeria.
Linha 9	Mostra a galeria.
Linha 10	Restabelece o texto da dica para o usuário.
Linha 12	Aqui começa a ação logo que a página é carregada e o usuário clica um thumbnail. Amplia a imagem clicada, alterando seus atributos para largura e altura, e revela usando o efeito `slideDown()`.
Linha 13	Altera o texto da dica ao usuário.

Observe, na figura 12.6, o efeito final da ampliação em uma tomada no momento em que a ampliação estava sendo processada.

Figura 12.6 – Galeria, imagem em ampliação.

Este script apresenta um comportamento indesejado que precisa ser corrigido. Para visualizar tal comportamento, abra o `arquivo-12.3a.html` que contém o exemplo, clique um thumbnail para ampliar a imagem e, a seguir, também esta, enquanto está sendo ampliada, ou seja, antes de se completar a ampliação.

Você vai notar que o clique na imagem em movimento faz que o processo de ampliação se inicie sem ter sido completada a primeira ampliação e a partir daí o script falha completamente, desconfigurando a galeria. Para resolver esse comportamento, é necessário desabilitar o clique na imagem enquanto está sendo ampliada.

Em linguagem corrente, deseja-se o seguinte: enquanto a imagem estiver em movimento de ampliação, não execute a função disparada por um clique nela, ou, ainda, por negação de condição: execute a função somente se a foto não estiver sendo animada. Usando a condição de negação e traduzindo para linguagem jQuery, tem-se:

```
if (!$('foto').is(':animated')) {//permitir clique para executar a função}
```

Complementarmente, será necessário desabilitar o cursor-padrão indicando link (mãozinha) durante a animação, sendo o script:

```
if ($('.foto').is(':animated')) {($('.foto').css('cursor', 'default')};
```

Finalmente, o script com os acréscimos necessários para corrigir o comportamento estranho detectado no script anterior:

```
1.  <script type="text/javascript">
2.  $(document).ready(function() {
3.      $('.foto').each(function() {
4.          $(this).click(function(event) {
5.          event.preventDefault();
6.          if (!$(this).is(':animated')) {
7.              $('.foto').hide();
8.              if ($(this).attr('width') > 100) {
9.                  $(this).attr({width:100, height:75});
10.                 $('.foto').show();
11.                 $('.abre-fecha').text('Clique para ampliar');
12.             } else {
13.                 $(this).attr({width:500, height:375}).slideDown(1500);
14.                 $('.abre-fecha').text('Clique para fechar');
15.                 }
16.             }
17.         if ($('.foto').is(':animated')) {$('.foto').css('cursor', 'default')};
            });
18.     });
19. });
20. </script>
```

[arquivo-12.3.1b.html]

Linha(s)	Descrição
Linha 11	Reduz a opacidade da imagem ampliada para 30%.
Linha 12	Coloca a imagem ampliada dentro da tela.
Linha 13	Anima a imagem ampliada aumentando sua opacidade de 30% para 100% em um intervalo de tempo de 2 segundos.

12.4 Slide-show

Nesta forma de apresentação, a galeria de imagens é construída sem auxílio de thumbnails, sendo apresentada inicialmente uma imagem em seu tamanho natural que será substituída pela seguinte com o uso de um efeito de transição. O método de substituição das imagens pode ser automático ou por interação do usuário com um mecanismo de navegação.

Primeira etapa

Nesta etapa, você irá retirar os thumbnails e apresentar inicialmente na página uma das imagens em seu tamanho ampliado. Uma dica ao usuário irá instruí-lo a clicar a imagem apresentada. A cada clique, haverá a troca da imagem em uma apresentação do tipo slide-show.

A marcação HTML e as CSS são as da página básica e o script para essa proposta é mostrado a seguir:

```
1.  <script type="text/javascript">
2.  $(document).ready(function() {
3.      $('.abre-fecha').text('Galeria: clique a foto');
4.      $('#galeria')
5.      .css({
6.          position:'relative',
7.          width:'500px',
8.          height:'375px'
9.      });
10.     $('.foto').attr({width:500, height:375})
11.     .css({position:'absolute', left:0, zIndex:1})
12.         .click(function(event) {
13.             event.preventDefault();
14.             $(this).slideUp(1500, function() {
15.             $(this).show().prependTo('#galeria').css('cursor', 'pointer');
16.             });
17.         });
18. });
19. </script>
```

[arquivo-12.4a.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Troca o texto da dica ao usuário. Note que na marcação o texto é: "Clique para ampliar", que é apropriado para a galeria e funciona sem o script.
Linhas 4 a 9	Dimensiona o `div` container de todas as imagens para uma largura e altura iguais às da imagem ampliada. O posicionamento relativo destina-se a criar um contexto de posicionamento, pois será utilizado `z-index` para colocar uma imagem sobre a outra no container e essa propriedade CSS só funciona para elementos posicionados. Você verá adiante que cada imagem receberá posicionamento absoluto dentro do container.
Linha 10	Amplia as imagens para seu tamanho natural, pois na marcação estão reduzidas. Este método sobrescreve as dimensões da marcação.
Linha 11	Posiciona cada imagem dentro do container de forma absoluta, no mesmo lugar, e atribui a todas elas o mesmo `z-index`. De acordo com as Recomendações para CSS, elementos com mesmo `z-index` são empilhados, levando-se em conta a ordem em que aparecem na marcação. O critério é que o elemento que se segue recebe maior `z-index`. Neste caso, as imagens foram marcadas de `1.jpg` até `10.jpg` e, consequentemente, o maior `z-index` será da imagem `10.jpg` e o menor, da imagem `1.jpg`. Assim, a imagem visível quando a página é carregada é a última imagem marcada. A sequência de visibilidade do slide-show será da última para a primeira imagem marcada. Pode-se facilmente alterar essa sequência fazendo um loop pelas imagens com o método `each()` e um contador. Esta solução será um bom exercício para você treinar seus conhecimentos.
Linha 12	Define uma função a ser executada quando o usuário clica uma imagem.
Linha 13	Desabilita a ação-padrão do navegador, impedindo que ele siga o link para o arquivo da imagem.
Linha 14	Esconde a imagem com efeito de animação `slideUp()` e define uma função a ser executada ao final da animação. Isto irá revelar a imagem que se segue no empilhamento.
Linha 15	Mostra a imagem que acaba de ser escondida e move-a para o início da pilha. Com isso, adquire o `z-index` mais baixo e só irá aparecer na segunda passada do slide-show. A imagem foi movida, mas o elemento `a` que era seu container não foi junto, então será necessária a declaração CSS `cursor pointer` para a segunda passada do slide-show. Note, ainda, que ao declarar `slideUp()` para a imagem, esta desaparece e permanece invisível. Por isso, utiliza-se o método `show()` para restabelecer a visibilidade da imagem.

Figura 12.8 – Galeria, navegação por botões.

É necessário acrescentar algumas regras CSS na folha de estilos básica da página que são mostradas a seguir.

▶ **CSS**:

```
#galeria a, #galeria a:visited {
   text-decoration:none;
   color:#333;
   padding:2px 6px;
   background:#999;
   margin-right:2px;
   }
#galeria a:hover, #tudo #galeria a.corrente {
   color:#999;
   background:#ccc;
   }
```

Essas regras CSS destinam-se a estilizar os botões de navegação. Note que se criou a classe `corrente` para ser inserida no botão clicado com a finalidade de destacá-lo, usando regras CSS apropriadas, e assim fornecer ao usuário a informação sobre qual imagem está sendo visualizada.

▶ jQuery

```
1.   <script type="text/javascript">
2.   $(document).ready(function() {
3.      $('.abre-fecha').hide();
4.      $('.foto').each(function(i) {
5.         $(this).replaceWith('<span >' + (i+1) + '</span>')
6.         $('#galeria').css('textAlign', 'center');
7.      });
8.      $('<div id="tela"><img src="1.jpg" alt="Descrição da imagem" /></div>')
9.      .insertBefore('#galeria')
10.     $('#galeria a').click(function(event) {
11.        event.preventDefault();
12.        $('#galeria a').removeClass('corrente');
13.        $(this).addClass('corrente');
14.        $('#tela img').attr('src', $(this).text() + '.jpg')
15.        .css('opacity', '0.3').animate({opacity:1}, 1500);
16.     });
17.  });
18.  </script>
```

 [arquivo-12.4c.html]

Código comentado:

Linha(s)	Descrição
Linha 3	Esconde a dica ao usuário, pois só é necessária quando JavaScript estiver desabilitado no navegador, não sendo necessária nesta solução.
Linha 4	Inicia um loop por todos os elementos `img` que marcam as fotos da galeria.
Linha 5	Substitui cada elemento imagem encontrado por um elemento `span` contendo um número. Serão tantos elementos `span` quantos forem as imagens da galeria numerados de 1 (um) até o número correspondente à quantidade de imagens encontrada. Com isso, criam-se os botões de navegação.
Linha 6	Esta regra CSS centraliza o conjunto de botões de navegação em relação à imagem ampliada.
Linha 8	Cria-se o elemento `div#tela` com dimensões iguais às da imagem ampliada. Veja, na folha de estilos da página, as regras CSS para esse elemento `div`.
Linha 9	Inseriu-se o elemento `div` criado acima dos botões de navegação. Veja na marcação da página que os botões criados estão contidos no elemento `div#galeria`.
Linha 10	Define-se uma função a ser executada quando for clicado um elemento `a` da navegação horizontal numérica.

Linha(s)	Descrição (cont.)
Linha 12	Remove-se a classe corrente que destaca a imagem clicada. Este método entrará em ação a partir do segundo clique em um número da navegação.
Linha 13	Insere-se a classe `corrente` para destacar o número clicado.
Linha 14	Indica-se o caminho para a imagem clicada usando a troca do atributo `src`.
Linha 15	Diminui-se para 30% a opacidade da imagem a revelar e, a seguir, anima-se a imagem fazendo sua opacidade aumentar para 100%.

Quarta etapa

Há situações nas quais se deseja fornecer um texto descritivo da imagem servindo como legenda. Nesta etapa, você irá estudar como apresentar uma legenda para cada imagem da galeria desenvolvida na etapa anterior.

Posicione a legenda, escrita sobre um fundo de cor sólida e transparente, sobre a própria imagem, conforme mostra a figura 12.9.

Figura 12.9 – Galeria com legenda.

A legenda deverá constar da marcação HTML da página básica e será constituída pelo texto do atributo `alt` de cada imagem. Isto é, se você pretende adotar essa solução em seu projeto, escreva a legenda para cada imagem no atributo em questão e o script irá extraí-la de lá.

O posicionamento da legenda é controlado por regras CSS. Neste caso, você irá definir posicionamento relativo para o container da imagem e posicionamento absoluto para a legenda. Assim fazendo, controlará com regras CSS o posicionamento da legenda sobre a imagem. As regras CSS a acrescentar na folha de estilos do script desenvolvido na etapa anterior são mostradas a seguir.

▶ **CSS**:

```
#tela {
   position:relative;
   width:500px; height:375px; margin:0 auto 5px auto; border:4px solid #999;}
#galeria a, #galeria a:visited {text-decoration:none; color:#333; padding:2px 6px;
   background:#999; margin-right:2px;}
#galeria a:hover, #tudo #galeria a.corrente {color:#999; background:#ccc;}
#tela  p.legenda {position:absolute; width:480px; left:0; bottom:0;
   font-size:14px; font-weight:bold;
   padding:2px 10px; border-top:1px dotted #fff; background:#000; margin:0;}
```

A legenda será marcada dentro de um parágrafo ao qual se atribuiu a classe denominada `legenda`. O parágrafo será posicionado de forma absoluta, com o uso de coordenadas `left` e `bottom`, dentro do elemento `div#tela` que contém a imagem. Aumentam-se o peso e tamanho da fonte, acrescentam-se `padding` e `border` para melhorar a apresentação do texto e declara-se margem inferior igual a 0 (zero) para o parágrafo, com o objetivo de evitar que a sobreposição de margens desloque o texto para cima em navegadores que não o Internet Explorer.

A seguir, veja os acréscimos e modificações no script anterior para obter o efeito proposto nesta etapa:

```
1.      <script type="text/javascript">
2.      $(document).ready(function() {
3.         $('.abre-fecha').hide();
4.         $('.foto').each(function(i) {
5.            $(this).replaceWith ('<span title="' +  $(this).attr('alt') +'">' +
                 (i+1) + '</span>');
6.            $('#galeria').css('textAlign', 'center');
7.         });
8.         $('<div id="tela"><img src="1.jpg" alt="Descrição da imagem" /></div>')
9.         .insertBefore('#galeria')
```

CAPÍTULO 13
Plug-ins

A biblioteca jQuery é um software de código aberto e extensível. Isto significa que qualquer um, com conhecimento da linguagem de programação JavaScript, pode modificar, acrescentar ou retirar funcionalidades de seu código original para satisfazer necessidades de seu projeto.

É esse caráter de extensibilidade que possibilita a criação de plug-ins, códigos escritos por terceiros, em linguagem JavaScript, que podem utilizar as regras de sintaxe da biblioteca e destinam-se a simplificar scripts que, com o uso das funcionalidades nativas, seriam mais complexos.

Este capítulo destina-se a mostrar como funcionam os plug-ins para a biblioteca jQuery. Você estudará como instalar e configurar os parâmetros de dois plug-ins escolhidos entre centenas existentes. Para uma listagem completa dos plug-ins disponíveis, consulte `http://plugins.jquery.com/`.

13.1 Introdução

13.1.1 Plug-ins de terceiros

Para fins de estudo, a instalação e o uso de um plug-in serão divididos em três fases distintas, a saber:

- primeira fase: fazer o download do plug-in.
- segunda fase: linkar os arquivos que fazem o plug-in funcionar na página em que será necessário.
- terceira fase: escrever o código com os parâmetros requeridos pelo plug-in.

O autor do plug-in, ao disponibilizá-lo para uso público, em geral cria uma página ou algumas páginas contendo informações sobre o plug-in que fornecem sua descrição e finalidade, instruções para instalação, exemplos de aplicação, parâmetros de configuração e informações relacionadas. De posse dessas informações, você terá condições de instalar e fazer funcionar os plug-ins.

13.1.2 Plug-ins nativos

Alguns plug-ins destinados a criar funcionalidades para a interface do usuário fazem parte nativamente da biblioteca jQuery e constam de sua documentação oficial. Tais plug-ins foram reunidos em uma seção da biblioteca denominada jQuery/UI e atualmente estão divididos em três categorias como mostrado a seguir:

- **Plug-ins nativos para interação**

Plug-in	Descrição
Draggables	Cria itens que podem ser arrastados com o uso do mouse.
Dropables	Cria alvos para soltar objetos arrastados.
Sortable	Manipula e ordena listas com o mouse.
Seletables	Seleciona itens com o mouse.
Resisables	Redimensiona objetos com o mouse.

- **Widgets**

Plug-in	Descrição
Accordion	Cria painéis retráteis para conteúdos.
Datepicker	Cria um calendário para selecionar datas.
Dialog	Cria uma caixa de diálogo flutuante.
Slider	Cria cursor deslizante.
Tab	Cria navegação por abas.

- **Effects**: reúne uma série de diferentes efeitos e a divisão formal em categorias não tem valor prático. Assim, será apenas citado para que você saiba de sua existência.

13.2 Plug-in jCarousel

Este plug-in foi desenvolvido por Jan Sorgalla e sua documentação encontra-se hospedada em `http://sorgalla.com/jcarousel/`. Segundo definição do autor, o plug-in destina-se a manipular horizontal ou verticalmente uma lista de itens. Os itens da

lista podem ser conteúdos estáticos ou dinâmicos carregados via AJAX ou com uso de outra programação qualquer. A manipulação dos conteúdos se faz com rolagem nos sentidos adiante e para trás, podendo ser animada ou não.

Trata-se de um plug-in extremamente útil para construção de galerias de imagens. O autor classifica o plug-in em três grupos conforme relacionados a seguir:

- Conteúdos estáticos
  - Carrossel simples vertical e horizontal.
  - Carrossel com scroll automático.
  - Carrossel com o uso de funções callback.
  - Carrossel com controles externos.
  - Carrossel com personalização da imagem de partida.
  - Vários carrosséis em uma página.
- Conteúdos dinâmicos
  - Carrossel com conteúdo carregado via JavaScript.
  - Carrossel com conteúdo carregado via AJAX.
  - Carrossel com conteúdo carregado via AJAX de um script PHP.
  - Carrossel com conteúdo carregado via AJAX com imagens hospedadas no Flickr (stream e API).
- Especiais
  - Carrossel circular.
  - Carrossel para rolagem de textos.
  - Carrossel elástico.
  - Carrossel e Thickbox 3.
  - Carrossel com efeito de animação personalizado.

Como se pode notar, as possibilidades de uso são inúmeras e não serão detalhadas todas elas porque o objetivo deste capítulo é relatar a existência dos plug-ins e mostrar os princípios de instalação e uso deles. Assim, você estudará detalhadamente alguns exemplos do grupo conteúdos estáticos e o uso especial com Thickbox 3.

13.2.1 Instalação

A instalação do plug-in consiste tão somente em fazer o download dos arquivos, descompactá-los, hospedar a pasta-raiz do plug-in e linkar os arquivos requeridos na página onde você pretende instalá-lo.

Visite `http://sorgalla.com/jcarousel/` e faça o download do arquivo disponível nas versões `jcarousel.tar.gz` ou `jcarousel.zip`. Descompacte o arquivo e hospede a pasta-raiz denominada `jcarousel`. A pasta-raiz contém várias subpastas e arquivos destinados a criar os diferentes efeitos a que o plug-in se propõe. Caso você saiba exatamente o que está fazendo, poderá deletar pastas e/ou arquivos não utilizados em seu projeto, mas se tiver dúvidas, mantenha a pasta-raiz do plug-in com todos seus conteúdos intactos. Tendo descompactado e hospedado o plug-in, tenha agora uma visão geral de seus componentes.

Na pasta `examples`, existem vários arquivos HTML contendo exemplos de todas as possibilidades de criação do plug-in.

A pasta `lib` contém as versões normal, compacta e codificada do script do plug-in. Uma dessas três versões deverá ser linkada ao documento que contém o carrossel. O critério de escolha de uma delas é o mesmo adotado para a linkagem da biblioteca propriamente dita. A versão mais compacta e que demanda menos tempo de carregamento é a do arquivo `jquery.jcarousel.pack.js`. Essa pasta contém ainda o arquivo de estilização geral do carrossel denominado `jquery.jcarousel.css`, que deverá ser linkado à página do carrossel, e uma subpasta denominada `thickbox` contendo os arquivos necessários à obtenção do efeito thickbox no carrossel.

A pasta `skins` contém duas subpastas denominadas `ie7` e `tango`. Essas subpastas contêm a folha de estilos e imagens necessárias para duas opções de apresentação visual, ou temas, do carrossel. Você pode personalizar a apresentação criando seu tema em uma subpasta com o nome do tema e nela inserir uma folha de estilo e imagens personalizadas. Se você pretende personalizar um skin, use como ponto de partida uma das duas folhas de estilo para um dos temas-padrão, mantendo o nome dos seletores, e grave-a com o nome `skin.css`.

Assim, toda página que contiver qualquer uma das versões de instalação do carrossel deverá ser linkada conforme mostrado a seguir:

```
1.    <head>
2.    ...
3.    <script type="text/javascript" src="../jquery-1.2.6.js"></script>
4.    <script type="text/javascript" src="jcarousel/lib/jquery.jcarousel.pack.js">
         </script>
```

```
6.     <script type="text/javascript">
7.        jQuery(document).ready(function() {
8.           jQuery('#chorizontal').jcarousel();
9.           jQuery('#cvertical').jcarousel({
10.             vertical:true
11.          });
12.       });
13.    </script>
```

 [arquivo-13.3a.html]

Código comentado:

Linha(s)	Descrição
Linha 1	Link para a biblioteca jQuery.
Linha 2	Link para o plug-in jCarousel. Este link deverá ser colocado após o link para a biblioteca.
Linha 3	Link para a folha de estilo geral.
Linha 4	Link para a folha de estilo do tema `ie7` que será utilizada na versão horizontal.
Linha 5	Link para a folha de estilo do tema `tango` que será usada na versão vertical.
Linhas 7 a 12	Container para scripts jQuery.
Linha 8	Sintaxe para o carrossel horizontal. Basta passar como parâmetro o valor do `id` do elemento `ul` que contém as imagens para o carrossel. Veja, na marcação HTML mostrada anteriormente, que o `id` adotado foi `chorizontal`.
Linhas 9 e 10	Sintaxe para o carrossel vertical. Basta passar como parâmetro o valor do `id` do elmento `ul` que contém as imagens para o carrossel e a seguir definir `vertical:true`. Veja na marcação HTML mostrada anteriormente, que o `id` adotado foi `cvertical`.

O resultado final é mostrado na figura 13.1.

Para controlar a velocidade da rolagem e o número de imagens a rolar de cada vez, use os parâmetros destacados a seguir:

```
jQuery(document).ready(function() {
   jQuery('#chorizontal').jcarousel({
      scroll:2,          // número de imagens a rolar de cada vez
      animation:1800     // velocidade da rolagem em milissegundos

   });
});
```

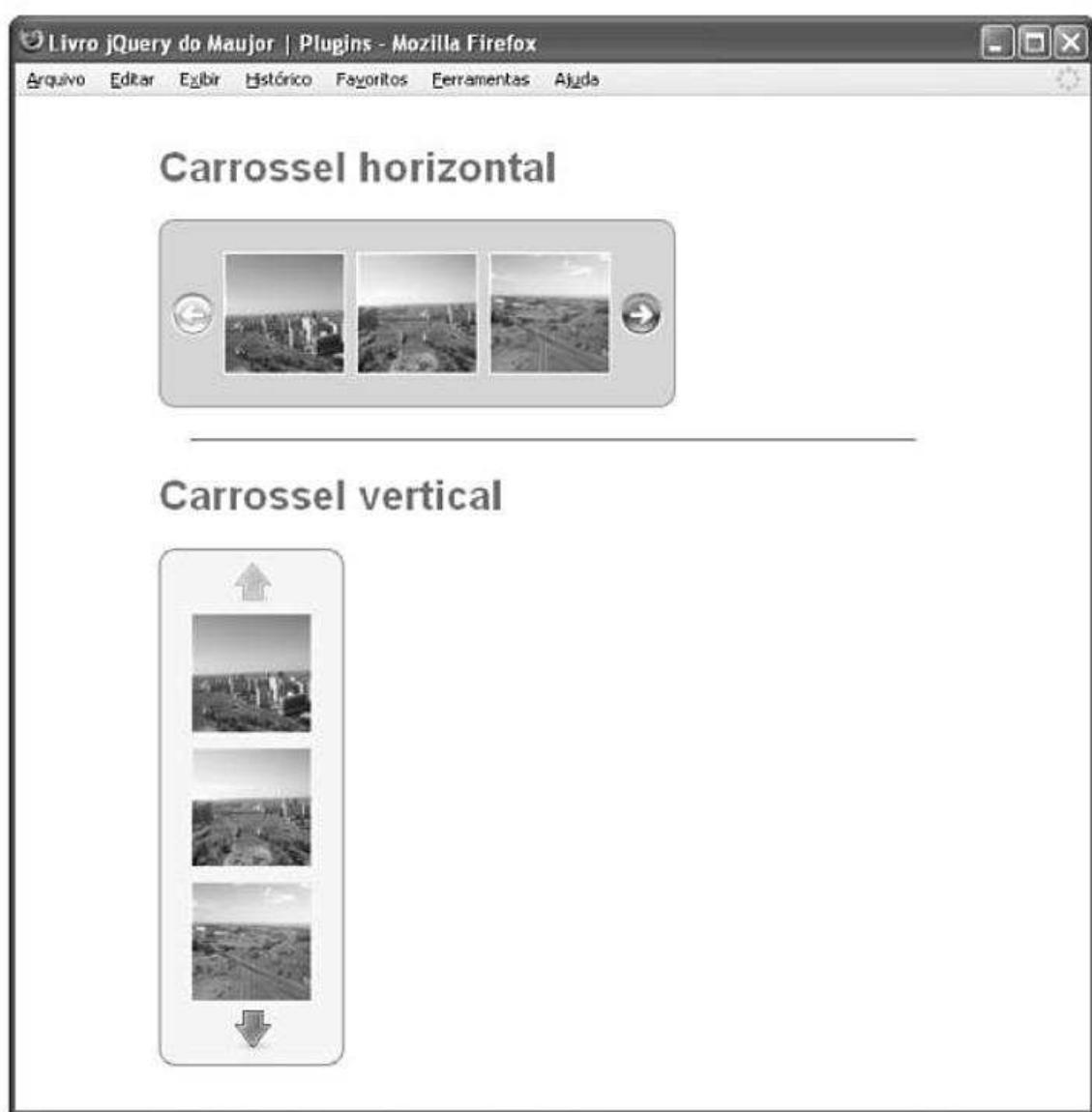



Figura 13.1 – Carrossel simples.

13.4 Carrossel com scroll automático

Nesta versão, o carrossel executa a movimentação das imagens automaticamente, em um intervalo de tempo configurável. Ao atingir a visualização das últimas imagens, a rolagem volta instantaneamente ao início (configurável como se verá adiante), repetindo-se o ciclo indefinidamente. São fornecidas as setas de comando manual e caso o usuário clique uma delas, a rolagem automática cessa e o carrossel passa a funcionar sem ela, até que se recarregue a página. Se o usuário coloca o ponteiro do mouse sobre uma imagem, cessa o scroll automático até que ele retire o ponteiro de cima da imagem.

Código comentado:

Linha(s)	Descrição
Linha 1	Link para a biblioteca jQuery.
Linha 2	Link para o plug-in jCarousel. Este link deverá ser colocado após o link para a biblioteca.
Linha 3	Link para a folha de estilo geral.
Linha 4	Link para a folha de estilo do tema `ie7` que será usada neste exemplo.
Linhas 5 a 18	Contém a definição de uma função-padrão do plug-in denominada pelo autor de `mycarousel_initCallback(carousel)`. Deve ser inserida na seção `head` do documento exatamente como mostrada no script. Copie e cole, retirando da documentação oficial do plug-in ou do arquivo HTML deste exemplo.
Linhas 7 a 12	Desabilita o scroll automático quando o usuário clica uma das setas para scroll manual.
Linhas 13 a 17	Para e reinicia o scroll automático quando o usuário aponta ou retira o ponteiro do mouse de cima da imagem.
Linhas 19 e 25	Container para scripts jQuery.
Linha 20	Passagem do parâmetro valor do `id` escolhido para o elemento `ul` de sua marcação.
Linha 21	Aqui você define o tempo em segundos entre cada scroll automático. Escolheu-se 1 segundo para fins de demonstração.
Linha 22	Aqui você controla o reinício do ciclo automático. O valor-padrão ou se não for especificado um valor, o scroll automático irá perdurar por um ciclo somente. Os valores possíveis são `last`, `first`, `both`, `null` (padrão). Para maiores detalhes, consulte `http://sorgalla.com/projects/jcarousel/#Configuration`.
Linha 23	Chama a função definida anteriormente.

Talvez você queira desabilitar os botões para scroll manual. Neste caso, acrescente o seguinte no script anterior:

```
jQuery(document).ready(function() {
    jQuery('#cscroll').jcarousel({
        auto:1,
        wrap:'last',
        buttonNextHTML:null, // Desabilita o botão: próximo
        buttonPrevHTML:null, // Desabilita o botão: anterior

        initCallback:mycarousel_initCallback
    });
});
```

No arquivo que ilustra esse exemplo são apresentadas duas versões conforme mostrado na figura 13.2.

Figura 13.2 – Carrossel com scroll automático.

13.5 Carrossel com comandos externos

Para esta versão, você deverá incluir uma folha de estilo para os comandos, uma marcação para a navegação por botões e a marcação de um menu `select` para a escolha de opções, tudo conforme mostrado a seguir.

Existe uma folha de estilo-padrão que controla a apresentação dos comandos externos e deverá ser incorporada ou linkada à página que contém o carrossel. Você pode personalizar a apresentação alterando as regras de estilo na citada folha sem, contudo, alterar o nome dos seletores CSS. Veja a seguir a folha de estilo para esta versão do carrossel:

```
<style type="text/css">
/**
 * Folha de estilos para os controles do carrosel.
 */
.jcarousel-control {margin-bottom:10px; text-align:center;}
.jcarousel-control a {font-size:75%; text-decoration:none; padding:0 5px;
   margin:0 0 5px 0; border:1px solid #fff; color:#eee;
   background-color:#4088b8; font-weight:bold;}
```

```
.jcarousel-control a:focus,
.jcarousel-control a:active {outline:none;}
.jcarousel-scroll {margin-top:10px; text-align:center;}
.jcarousel-scroll form {margin:0; padding:0;}
.jcarousel-scroll select {font-size:75%;}
#mycarousel-next,
#mycarousel-prev {cursor:pointer; margin-bottom:-10px;
   text-decoration:underline; font-size:11px;}
</style>
```

A marcação HTML é mostrada a seguir:

```
<div id="mycarousel" class="jcarousel-skin-ie7">
   <div class="jcarousel-control">
      <a href="#">1</a>
      <a href="#">2</a>
      ...
      <a href="#">9</a>
      <a href="#">10</a>
   </div>

<ul class="jcarousel-skin-ie7">
<li><img src="1_s.jpg" alt="Descrição da foto" width="75" height="75" /></li>
...
<li><img src="10_s.jpg" alt="Descrição da foto" width="75" height="75" /></li>
</ul>

<div class="jcarousel-scroll">
   <form action="">
   <a href="#" id="mycarousel-prev">&laquo; Anterior</a>
      <select>
         <option value="1">Scroll 1 item por click</option>
         <option value="2">Scroll 2 items por click</option>
         <option value="3">Scroll 3 items por click</option>
         <option value="4">Scroll 4 items por click</option>
         <option value="5">Scroll 5 items por click</option>
      </select>
   <a href="#" id="mycarousel-next">Próxima &raquo;</a>
   </form>
</div>
</div>
```

Crie a marcação para o carrossel em três blocos de código como mostrado. Os três blocos devem ter como container, obrigatoriamente, um `div` com `id="mycarousel"` e `class="jcarousel-skin-nomedotema"`. Não altere esses nomes identificadores.

O primeiro bloco de código tem como container um div com `class="jcarousel-controls"` e contém tantos links mortos quantas forem as imagens constantes do carrossel. Neste exemplo, há dez imagens.

O segundo bloco de código é o elemento `ul` com as imagens. Esse bloco difere dos já estudados para as versões anteriores porque aqui não há necessidade de atribuir um `id` para o elemento `ul`, bastando definir uma classe com o nome do tema.

O terceiro bloco de código deve estar contido em um elemento `div` com `id="jcarousel-scroll"` e marca um menu `select` para escolha do número de imagens a ser rolada em cada clique para avançar ou recuar.

O script que faz funcionar esta versão é o seguinte:

```
<script type="text/javascript">
function mycarousel_initCallback(carousel) {
    jQuery('.jcarousel-control a').bind('click', function() {
        carousel.scroll(jQuery.jcarousel.intval(jQuery(this).text()));
        return false;
    });
    jQuery('.jcarousel-scroll select').bind('change', function() {
        carousel.options.scroll = jQuery.jcarousel.intval(this.options[this.
            selectedIndex].value);
        return false;
    });
    jQuery('#mycarousel-next').bind('click', function() {
        carousel.next();
        return false;
    });
    jQuery('#mycarousel-prev').bind('click', function() {
        carousel.prev();
        return false;
    });
};
jQuery(document).ready(function() {
    jQuery('#mycarousel').jcarousel({
        scroll:1,
        initCallback:mycarousel_initCallback,
        buttonNextHTML:null,
        buttonPrevHTML:null
    });
});
</script>
```

[arquivo-13.5a.html]

Veja, na figura 13.4, o carrossel com efeito thickbox.

Figura 13.4 – Carrossel com efeito thickbox.

13.7 Plug-in jQuery Accordion

Este plug-in foi desenvolvido por Jörn Zaefferer e sua documentação encontra-se hospedada em `http://bassistance.de/jquery-plugins/jquery-plugin-accordion/`. Trata-se de um plug-in para obter o conhecido efeito sanfona e pode ser usado para esconder e revelar tanto conteúdos textuais como listas de links. No caso de listas de links, o plug-in é útil no desenvolvimento de mecanismos de navegação.

Admite várias estruturas de marcação HTML, cada uma de acordo com a natureza dos conteúdos a revelar e esconder. As folhas de estilo para apresentação das diversas versões criadas devem ser desenvolvidas de acordo com a necessidade de cada projeto, contudo a documentação do plug-in em sua seção de demonstração desenvolve quatro exemplos de uso do plug-in com respectivas folhas de estilo que podem servir de base para a criação de folhas de estilo personalizadas.

Você irá estudar duas variantes de marcação para uso prático do plug-in: uma para conteúdos textuais e outra para um mecanismo de navegação.

13.7.1 Instalação

A instalação do plug-in consiste tão somente em fazer o download dos arquivos, descompactá-los, hospedar a pasta-raiz do plug-in e linkar os arquivos requeridos na página onde você pretende instalá-lo.

Visite http://bassistance.de/jquery-plugins/jquery-plugin-accordion/ e faça o download do arquivo disponível denominado jquery.accordion.zip. Descompacte o arquivo e hospede a pasta-raiz denominada jquery-accordion. A pasta-raiz contém várias subpastas e arquivos destinados a criar os diferentes efeitos a que o plug-in se propõe. Caso você saiba exatamente o que está fazendo, poderá deletar pastas e/ou arquivos não utilizados em seu projeto, mas se tiver dúvidas, mantenha a pasta-raiz do plug-in com todos os seus conteúdos intactos. Tendo descompactado e hospedado o plug-in, veja melhor seus componentes.

Essa pasta contém ainda as versões normal, compacta e codificada do script do plug-in. Uma dessas três versões deverá ser linkada ao documento que contém o efeito sanfona. O critério de escolha de uma delas é o mesmo adotado para a linkagem da biblioteca propriamente dita. A versão mais compacta e que demanda menos tempo de carregamento é a do arquivo jquery.jcarousel.pack.js.

Na pasta demo, os arquivos index.html e nested.html contêm exemplos de todas as possibilidades de criação do plug-in. Você irá estudar com detalhes os dois primeiros exemplos contidos no arquivo index.html.

A pasta lib contém arquivos plug-ins complementares a serem usados em diferentes situações, conforme se verá adiante. Nessa pasta, encontra-se um plug-in denominado jquery.dimensions.js, que, a partir da versão 1.2.6 da biblioteca, foi incorporado a ela, não havendo mais necessidade de seu uso em separado a partir dessa versão.

A pasta test contém scripts não relacionados diretamente ao desenvolvimento do efeito sanfona em sites.

Assim, toda página que contenha uma das versões de instalação do efeito sanfona deverá ser linkada conforme mostrado a seguir:

```
1.    <head>
2.    ...
3.    <script type="text/javascript" src="../jquery-1.2.6.js"></script>
4.    <script type="text/javascript" src="jquery-accordion/jquery.accordion.js"></script>
5.    <script type="text/javascript" src="jquery-accordion/xxxxxxxx.js"></script>
6.    </head>
```

```
10.         <a>Instalação do plug-in</a>
11.             <div>
12.                 ...conteúdo...
13.             </div>
14.     </div>
```

Código comentado:

Linha(s)	Descrição
Linhas 1 a 15	Um div container para toda a marcação. A esse container deve ser atribuído um id com nome de sua livre escolha, no exemplo adotado, id="a_um", e uma classe com nome basic. Essa classe será o seletor para a folha de estilo do efeito.
Linhas 2 a 5	Um bloco de código que se repete tantas vezes quantos forem os títulos dos conteúdos a revelar. Note que esse bloco se repete duas vezes no exemplo, linhas 6 a 9 e 10 a 14.
Linha 2	A marcação do título a revelar deverá ser feita com o elemento a. Nas linhas 6 e 10, estão os outros dois títulos.
Linhas 3 e 5	Um div container para os conteúdos a revelar e esconder.
Linha 4	Marcação para os conteúdos. Aqui você poderá usar qualquer elemento HTML requerido pela semântica do conteúdo marcado.

A folha de estilo-padrão, para esse efeito, retirada da documentação do plug-in é mostrada a seguir. Note que o container geral é um elemento div com a classe basic conforme mostrado na marcação anterior:

```
1.      .basic {width:260px; border:1px solid black;}
2.      .basic div {padding:10px;}
3.      .basic a {cursor:pointer; display:block; padding:5px; text-decoration:none;
            font-weight:bold;
            background:#00a0c6 url(jquery-accordion/demo/AccordionTab0.gif);
            border-top:1px solid #fff; border-bottom:1px solid #999;}
4.      .basic a:link:hover, .basic a:hover
            {background:#fff url(jquery-accordion/demo/AccordionTab2.gif);}
5.      .basic a.selected {background:#80cfe2
            url(jquery-accordion/demo/AccordionTab2.gif);}
```

Código comentado:

Linha(s)	Descrição
Linha 1	Define uma largura igual a 260px para o container e uma borda de 1px em volta dele.
Linha 2	Define um espaçamento de 10px em volta dos conteúdos a revelar. Note que se existir um elemento div na marcação dos conteúdos, esse espaçamento será herdado. Se não se desejar a herança, crie regra CSS apropriada para esses elementos.

Linha(s)	Descrição (cont.)
Linha 3	Estiliza os títulos. O exemplo na documentação usa uma imagem de fundo para compor o fundo dos títulos. Esta imagem está na pasta `demo` do plug-in.
Linha 4	Estilização para o estado mouse over nos títulos. A folha de estilo original usa somente o seletor `.basic a:hover`, omitindo o seletor `.basic a:link:hover`. Acrescentou-se este seletor para servir o navegador Internet Explorer que não reconhece o simples `a:hover` em elementos `a` não marcados com o atributo `href`, como é o caso da marcação para o exemplo. Aqui também é usada uma imagem de fundo.
Linha 5	Estilização para o título corrente usando uma imagem de fundo.

A sanfona simples admite duas versões. A primeira com altura constante para o `div` revelado, independentemente da quantidade de conteúdo. Tal altura é maior que a altura do `div` com mais quantidade de conteúdo. A outra versão é com altura do `div` de acordo com a quantidade de conteúdo. O parâmetro que diferencia as versões é `autoheight`. Declarando `autoheight:false` a altura será variável. O valor-padrão de `autoheight` é `true`, ou seja, nada declarando, a altura será fixa.

No exemplo, há duas versões em uma mesma página, conforme mostra a figura 13.5.

Figura 13.5 – Sanfona simples.

```
16.         <li><a class="head" href="#">CSS em ação</a>
17.             <ul>
18.             <li><a href="http://www.maujor...>Font interativo</a></li>

19.             <li><a href="http://www.maujor...>Text interativo</a></li>

20.             ...
21.             </ul>
22.         </li>
23.     </ul>
```

Código comentado:

Linha(s)	Descrição
Linhas 1 a 23	Um elemento para listas não ordenadas `ul` container para toda a marcação. A esse container deve ser atribuído um `id` com nome `navigation`. Esse `id` será o seletor para a folha de estilo do efeito.
Linhas 2 a 8	Um bloco de código que se repete tantas vezes quantos forem os títulos dos conteúdos a revelar. Note que esse bloco se repete duas vezes no exemplo, nas linhas 9 a 15 e 16 a 22.
Linha 2	A marcação do título a revelar deverá ser feita com o elemento `a` ao qual se atribuem uma classe denominada `head` e um atributo `href="#"`(link morto). Esse elemento `a`, por sua vez, deve estar contido em um elemento `li`. Nas linhas 9 e 16 estão os outros dois títulos. O citado elemento `li`, além de conter o título, é container para um elemento `ul` onde constam todos os links relacionados ao título. Trata-se de uma lista aninhada à lista container geral.

A folha de estilo-padrão para esse efeito, retirada da documentação do plug-in, é mostrada a seguir. Note que o container geral é um elemento `ul` com `id="navigation"` conforme mostrado na marcação anterior:

```
1.    #navigation {border:1px solid #5263AB; margin:0; padding:0; width:200px;}
2.    #navigation a.head {border:1px solid #ccc; background:#5263AB
          url(jquery-accordion/demo/collapsed.gif) no-repeat 3px 4px; color:#fff;
          display:block; font-weight:bold; margin:0; padding:0; text-indent:14px;
          text-decoration:none;}
3.    #navigation a.head:hover {color:#ff9;}
4.    #navigation a.selected {background-image:url(jquery-accordion/demo/expanded.gif);}
5.    #navigation a.current {background:#ff9;}
6.    #navigation ul {margin:0;padding:0;}
7.    #navigation li {list-style:none}
8.    #navigation li li a {color:#333; display:block; text-indent:10px;
          text-decoration:none;}
9.    #navigation li li a:hover {background:#ff9; color:#f00;}
```

```
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript" src="jquery-accordion/jquery.accordion.js"></script>
<script type="text/javascript" src="jquery-accordion/lib/jquery.easing.js"></script>
```

Para informações detalhadas sobre o plug-in `jquery.easing.js`, consulte o endereço `http://gsgd.co.uk/sandbox/jquery/easing/`.

O script para essa versão é mostrado a seguir:

```
1.   <script type="text/javascript">
2.      jQuery(document).ready(function() {
3.         jQuery('#navigation').accordion({
4.            active:false,
5.            header:'.head',
6.            event:'mouseover',
7.            fillSpace:false,
8.            animated:'easeslide'
9.            });
10.     });
11.  </script>
```

Código comentado:

Linha(s)	Descrição
Linha 4	Fecha o menu no início.
Linha 5	Nome da classe atribuída ao elemento `a` que marca os títulos.
Linha 6	Define o evento que abre o menu. Utilizou-se o valor `mouseover` no exemplo. Experimente mudar para `click` e observe a diferença.
Linha 7	Parâmetro para definir a altura do menu aberto. Usou-se `false` (este é o valor-padrão e não declará-lo surte o mesmo efeito) no exemplo para forçar a altura de acordo com o conteúdo revelado. O valor `true` mantém a altura constante independentemente do conteúdo revelado, fazendo o menu estender-se mais.
Linha 8	Interfere na aceleração da velocidade de animação. Utilizou-se o valor `easeslide` no exemplo. Experimente mudar para `bouceslide` e observe a diferença.

É somente isso. Após linkar o plug-in na página, criar uma folha de estilo para o efeito e desenvolver a marcação HTML como mostrado, basta declarar `jQuery('#um_nome_qualquer').accordion()` e alguns parâmetros, sendo *`um_nome_qualquer`* o nome que você escolheu para a `ul` container geral dos conteúdos a revelar com o uso do efeito sanfona.

Para maiores detalhes e outras versões de implementação, consulte a página oficial do plug-in localizada em `http://bassistance.de/jquery-plugins/jquery-plugin-accordion/` e a documentação no site jQuery em `http://docs.jquery.com/UI/Accordion/accordion`.

CAPÍTULO 14

Menu Maujor

Denomina-se menu Maujor o mecanismo de navegação do site `http://www.maujor.com` que usa a biblioteca jQuery para seu funcionamento. Não se trata de um mecanismo de navegação inédito nem de uma invenção exclusiva e única. Ao contrário, é sustentado por um script extremamente simples criado para incrementar a navegação que já existia no site, sem necessidade de alterar a marcação HTML.

14.1 Introdução

O menu do site do Maujor foi criado e expandido muito antes de a biblioteca jQuery ter se tornado uma realidade no universo do desenvolvimento de sites.

O menu passou por quatro fases. A primeira fase foi desde sua criação, quando começou com poucos links, até o momento em que a inserção de novos links passou a exigir uma coluna de navegação muito extensa. Nessa ocasião, iniciou-se a segunda fase, quando se optou por um menu do tipo pop-up lateral, em que os links para as matérias eram revelados quando o usuário passava o mouse sobre um título geral. A terceira fase baseou-se na solução adotada no redesign de um grande portal brasileiro e consistiu em reduzir a altura do container do menu; com o uso de regras CSS apropriadas, forçou-se o aparecimento de uma barra de rolagem vertical para tal container.

Passado algum tempo, houve algumas críticas ao sistema de navegação do site. Embora fossem esporádicas e longe de ser unanimidade, considerou-se o estudo de uma alternativa para o menu.

A quarta fase foi a implantação de um script jQuery que atualmente faz funcionar o menu. A aceitação foi total e houve muitos e-mails de leitores perguntando como funciona o menu. Foram tais solicitações que motivaram a escrita deste capítulo.

Neste capítulo, será dada ênfase especial, além do script jQuery, à marcação HTML e às regras CSS para a apresentação do menu. Como base para o exemplo a desenvolver, será adotado o menu atual do site, incluindo seus links funcionais.

14.2 HTML e CSS

Funcionalmente, trata-se de um menu hierárquico de dois níveis. O primeiro nível contém um título e sob ele estão os links para as matérias relacionadas a esse título.

O elemento HTML semanticamente correto para marcação de mecanismos de navegação são as listas HTML em suas três versões: lista ordenada `ol`, lista não ordenada `ul` e lista de definição `dl`. Optou-se pelo uso de listas não ordenadas para marcar o segundo nível do menu e o elemento `h3` para marcar o título ou primeiro nível. Outra solução seria usar lista de definição com o elemento `dt` para títulos e `dd` para o segundo nível. Tal opção de marcação será um exercício para você testar seu aprendizado.

Observe a seguir um trecho da marcação HTML do menu:

```
1.    <div id="menu">
2.       <h3>Destaques</h3>
3.          <ul >
4.             <li><a href="..." title="...">FAQ - CSS</a></li>
5.             <li><a href="..." title="...">QUIZ - CSS</a></li>
6.             <li><a href="..." title="...">Editor CSS do DWMX 2004</a></li>
7.             <li><a href="..." title="...">Aprenda CSS desde o inicio</a></li>
8.             <li>...</li>
9.          </ul>

10.      <h3>Fundamentos CSS</h3>
11.         <ul >
12.            <li><a href="..." title="...">Introdução às CSS</a></li>
13.            <li><a href="..." title="..."> Sintaxe CSS</a></li>
14.            <li><a href="..." title="...">Tipos de CSS</a></li>
15.            <li><a href="..." title="...">A propriedade CSS font</a></li>
16.            <li>...</li>
17.         </ul>

18.      <h3>Título primeiro nível</h3>
19.         <!-- Repete marcação anterior -->
20.   </div> <!-- fim do div#menu -->
```

Estilização inicial

Acrescente uma folha de estilos na página do menu para melhorar sua apresentação. Observe um primeiro conjunto de regras CSS conforme mostradas a seguir:

```
1.  <style type="text/css" media="all">
2.  #menu ul {
3.     list-style-type:none;
4.     margin:0;
5.     padding:0;
6.     }
7.  #menu li a {
8.     display:block;
9.     background:#d4dde4 url(seta.gif) no-repeat 98%;
10.    color:#333;
11.    border-left:5px solid #7d97ad;
12.    text-decoration:none;
13.    padding:3px 15px 3px 3px;
14.    margin-bottom:1px;
15.    }
16. #menu li a:hover {
17.    background:#dce3e9 url(seta.gif) no-repeat 98%;
18.    color:#999;
19.    border-left:5px solid #cad2dd;
20.    }
21. </style>
```

 [arquivo-14.2b.html]

Código comentado:

Linha(s)	Descrição
Linhas 2 a 6	Retira os marcadores das listas não ordenadas e os espaçamentos-padrão de `margim` e `padding` para listas. Esses espaçamentos causam uma tabulação à esquerda dos itens de lista que, além de variar de dimensão, de acordo com o navegador, são implementados ora pela propriedade `margin`, ora por `padding`. A finalidade desta regra CSS é igualar a renderização da lista para todos os navegadores.
Linha 7	Seletor para os links no estado inicial.
Linha 8	Elementos HTML `a` são elementos in-line e, como tais, ocupam a exata largura e altura necessárias a acomodar seus conteúdos. Isto faz que, em um link de texto, somente a área do texto seja clicável. O link em questão está inserido em um elemento `li` que tem uma forma retangular e na qual está inserido o texto do link. A declaração `display:block` nesta linha torna o elemento `a` nível de bloco, ocupando todo o seu container e fazendo que toda a área retangular do elemento `li` seja clicável.

Linha(s)	Descrição (cont.)
Linha 9	Define a cor de fundo azul-clara para o retângulo clicável e coloca uma seta decorativa no lado direito dele.
Linha 10	Elege a cor preta para o texto do link.
Linha 11	Define uma borda esquerda decorativa, na cor azul-escura, para o retângulo que contém o link.
Linha 12	Retira o sublinhado-padrão do texto do link.
Linha 13	Define os espaçamentos vertical e horizontal do texto do link em relação aos limites internos do retângulo que o contém.
Linha 14	Define uma `margin` inferior para cada retângulo container dos links, abrindo um espaço de 1px no qual será vista a cor de fundo definida para o elemento `div#menu`, criando a ilusão de uma linha de separação entre os links. Uma opção para essa regra CSS é declarar uma borda inferior de 1px na cor pretendida para a linha de separação, contudo, nesta opção, há uma junção das margens do último link aberto com a margem superior do título, resultando uma margem de 2px nesse local. Altere o arquivo e comprove você mesmo.
Linha 16	Regra CSS para o comportamento dos links quando o usuário passa o mouse sobre eles.
Linha 17	Fundo dos links em cor azul mais clara e redefinição da seta à direita. Se não tivesse declarado a seta como imagem de fundo, esta seria coberta pela nova cor e não apareceria na situação de mouse over.
Linha 18	Cor mais clara para o texto do link.
Linha 19	Cor mais clara para a borda esquerda.

A renderização do menu, nos navegadores Firefox 3 e Internet Explorer 6, com as primeiras estilizações é apresentada na figura 14.2. O efeito mouse over é mostrado no segundo link na tomada da imagem para o Firefox 3.

Bug no IE6 e anteriores

Este primeiro conjunto de regras de estilo causa a renderização do menu com um espaçamento não desejado, no navegador Internet Explorer 6 e anteriores, conforme mostra a figura 14.2. Este é um comportamento decorrente de uma das maiores causas das inconsistências de renderização no IE6 e anteriores, felizmente já corrigida no IE7. Para maiores informações, consulte `http://www.maujor.com/tutorial/haslayout.php`.

Conforme explicado na matéria publicada no link indicado no parágrafo anterior, para corrigir o bug mostrado, basta dar layout aos elementos `li` e `a`, acrescentando a seguinte regra CSS:

```
* html li, * html li a {height:1%;}
```

Figura 14.2 – Menu Maujor: estilização na primeira fase.

Dimensões do menu

O menu ocupa toda a janela do navegador e estende-se na vertical por todos os links de navegação, fazendo aparecer a barra de rolagem vertical. Defina uma largura para o menu, de modo que fique contido na coluna de navegação do site. Delimite, ainda, uma altura máxima e forneça uma barra de rolagem vertical para o menu. Adote uma largura de 200px e uma altura de 300px. Você deverá escolher essas dimensões de modo que se adaptem ao layout do site. Estilize os cabeçalhos `h3` que integram o primeiro nível do menu.

Observe as regras CSS a ser acrescentadas na folha de estilo inicial:

```
1.    body {
2.        font:11px/1.2 Arial, Helvetica, sans-serif;
3.        background:#d6e2e5; color:#666;
4.        padding:0; margin:0;
5.        }
```

```
6.     #menu {
7.         width:200px; height:300px;
8.         overflow:auto;
9.         margin:30px 0;
10.        background:#999;
11.        }
12.    #menu h3 {
13.        font-size:12px; cursor:pointer;
14.        line-height:18px;
15.        padding-left:20px;
16.        margin:1px 0;
17.        background:#cad2dd url(mais.gif) 3px center no-repeat;
18.        border-left:5px solid #7d97ad;
19.        }
20.    #menu h3.corrente {background:#cad2dd url(menos.gif) 3px center no-repeat;}
21.    * html li, * html li a {height:1%;}
```

 [arquivo-14.2c.html]

Código comentado:

Linha(s)	Descrição
Linha 1	Regras CSS gerais para a página que contém o menu.
Linha 7	Define as dimensões máximas do menu.
Linha 8	Cria barras de rolagem vertical e horizontal caso o conteúdo do menu ultrapasse as dimensões estabelecidas. Ainda que a dimensão vertical do menu que você está projetando seja menor que as dimensões estabelecidas na regra CSS, declare `overflow:auto` para o caso de expansão futura.
Linha 9	Define margens superior e inferior.
Linha 10	Cor de fundo do menu. Esta é a cor das linhas horizontais que separam os retângulos dos links.
Linha 12	Regras CSS para os cabeçalhos `h3` do primeiro nível do menu. Note a declaração para mudança do cursor indicativa de que o clique causará uma ação (no caso, abre os submenus) e também a definição de um sinal de mais (+) como uma imagem de fundo, indicando que há algo a abrir quando se clica o cabeçalho.
Linha 20	Regra CSS para colocar a imagem do sinal menos (-) no cabeçalho aberto.
Linha 21	Hack para o IE6 e anteriores conforme relatado anteriormente.

A renderização do menu, nos navegadores Firefox 3 e Internet Explorer 7, com os acréscimos na folha de estilo é apresentada na figura 14.3.

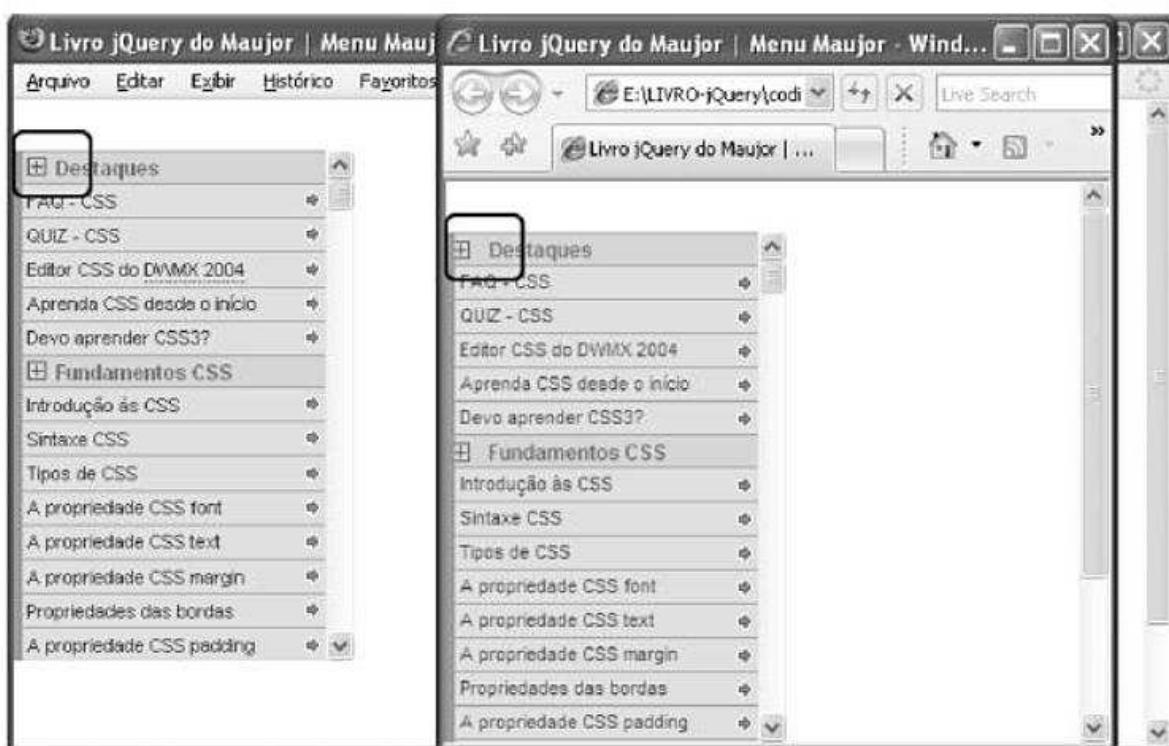



Figura 14.3 – Menu Maujor: estilização na segunda fase.

Bug no IE7 e anteriores

Com as novas regras CSS, teoricamente a estilização estaria terminada não fosse um bug nos navegadores Internet Explorer 7 e anteriores. Note, na figura 14.2, que se colocou em destaque, no primeiro cabeçalho, uma diferença no espaçamento da figura de fundo do sinal de mais (+) indicativo de subnível do menu.

Note, na linha 17 das regras de estilo mostradas anteriormente, que se definiu um total de 3px para a coordenada esquerda da imagem. O navegador Firefox colocou a imagem a 3px do limite direito da borda esquerda e o navegador Internet Explorer ignorou a borda e colocou a imagem a 3px do limite esquerdo do menu. Assim, para o IE, é necessário "puxar" a imagem para a direita de um valor igual a 5px, resultando em uma coordenada esquerda igual a 3px + 5px = 8px. A regra CSS que corrige o bug é mostrada a seguir:

```
* html #menu h3, * html #menu h3.corrente {  /* IE6 e anteriores */
   background-position:8px center;
   }
*:first-child+html #menu h3,
*:first-child+html #menu h3.corrente { /* IE7 */
   background-position:8px center;
   }
```

Linha(s)	Descrição (cont.)
Linha 6	Ao clicar um elemento `h3` do menu, altera-se a visibilidade do elemento que se segue. Veja, na marcação HTML, que se trata de um elemento `ul` que é o container do segundo nível para esse cabeçalho.
Linha 7	Esconde, com o uso do método `slideUp()`, os links do segundo nível que estejam abertos. Isto só tem efeito a partir do segundo clique num elemento `h3` quando já existe um subnível aberto.
Linha 8	Insere a classe `corrente` no cabeçalho clicado, fazendo que a imagem de fundo altere do sinal mais (+) para o sinal menos (-).
Linha 9	Retira a classe `corrente` do submenu que se fecha, alterando a imagem de fundo de um sinal menos (-) para o sinal mais (+).

Na figura 14.4, mostramos o efeito final com o primeiro conjunto de links aberto.

Figura 14.4 – Menu Maujor: efeito final.

APÊNDICE A

Seletores

A sintaxe dos seletores previstos nas Recomendações do W3C para as CSS 1, CSS 2.1 e também aqueles constantes dos estudos para a futura Recomendação das CSS 3 foi usada na criação dos seletores para a biblioteca jQuery. Assim, um perfeito entendimento de seletores CSS é indispensável para o bom desenvolvimento de scripts jQuery.

Este apêndice contém uma descrição geral dos seletores CSS e sua respectiva sintaxe jQuery. Apresenta, ainda, uma tabela na qual são relacionados os seletores CSS 3 e seus alvos na árvore do documento, empregados na biblioteca.

A.1 Seletor tipo

O seletor é a letra ou string que representa um elemento, na marcação HTML, conforme mostrado a seguir.

- Sintaxe CSS:

```
p {...}       /* o alvo são todos os parágrafos no documento */
h3 {...}      /* o alvo são todos os cabeçalhos nível três no documento */
code {...}    /* o alvo são todos os elementos code no documento */
```

- Sintaxe jQuery:

```
$('p')        // o alvo são todos os parágrafos no documento
$('h3')       // o alvo são todos os cabeçalhos nível três no documento
$('code')     // o alvo são todos os elementos code no documento
```

A.2 Seletor identificador único

O seletor é o valor do atributo `id` definido para um elemento, na marcação HTML, conforme mostrado a seguir.

- Sintaxe CSS:

```
#tudo {...}         /* o alvo é elemento com atributo id="tudo" */
#principal {...}   /* o alvo é elemento com atributo id="principal" */
```

- Sintaxe jQuery:

```
$('#tudo')          // o alvo é elemento com atributo id="tudo"
$('#principal')    // o alvo é elemento com atributo id="principal"
```

A.3 Seletor classe

O seletor é o valor do atributo `class` definido para um ou mais elementos, na marcação HTML, conforme mostrado a seguir.

- Sintaxe CSS:

```
.estrutura {...} /* o alvo são todos os elementos com atributo class="estrutura" */
.cor_um {...}   /* o alvo são todos os elementos com atributo class="cor_um" */
```

- Sintaxe jQuery:

```
$('.estrutura')  // o alvo são todos os elementos com atributo class="estrutura"
$('.cor_um')    // o alvo são todos os elementos com atributo class="cor_um"
```

A.3.1 Classificação dos seletores

Os seletores classificam-se em dois grandes grupos: seletores simples e seletores compostos.

A.3.1.1 Seletor simples

Seletor simples é aquele constituído de um só elemento, podendo ou não estar associado a uma classe, um `id` ou uma pseudoclasse. Veja a seguir exemplos de seletores simples.

- Sintaxe CSS:

```
p, p.um, p#principal
```

- Sintaxe jQuery:

```
$('p'), $('p.um'), $('p#principal')
```

- Sintaxe jQuery:

```
$('p[title]')
$('abbr[class="diferente"]')
$('div[id!="navegacao"]')
$('h1[class^="est"]')
$('div[class$="oico"]')
$('div[class*="ren"]')
```

A.3.1.2 Seletor composto

Seletor composto é aquele constituído pela combinação de dois ou mais seletores simples. A combinação entre seletores simples para criar um seletor composto segue uma sintaxe própria e é feita com o emprego de três sinais de combinação, como descrito na tabela A.1.

Tabela A.1 – Sinais de combinação

Sinal de combinação	Exemplo ilustrativo	Nota
Espaço em branco	div p em	Obrigatório usar um ou mais espaços entre seletores
Sinal de maior “>”	div > p ou div>p	Espaçamento facultativo entre seletores
Sinal de adição “+”	p + a ou p+a	Espaçamento facultativo entre seletores
Sinal til “~”	h1 ~ p	Espaçamento facultativo entre seletores

Árvore do documento

Para um sólido entendimento dos seletores compostos, é necessário conhecer a árvore do documento e sua terminologia. Considere a marcação a seguir:

```
...
<body>
    <div id="topo">
      <h1>Empresa</h1>
      <ul id="nav">
        <li><a href="#">Link 1</a></li>
        <li><a href="#">Link 2</a></li>
        <li><a href="#">Link 3</a></li>
      </ul>
    </div>
    <div id="principal">
      <h2>Título</h2>
      <p>...parágrafo <em>itálico</em>...</p>
      <h2>Título</h2>
```

A.4 Seletores avançados

Os seletores previstos nos estudos para a implementação das Recomendações do W3C para CSS 3 e que são usados na biblioteca jQuery são mostrados na tabela A.3.

Tabela A.3 – Seletores CSS 3

Seletor	Alvo
`E[att^="val"]`	Casa com qualquer elemento E cujo valor do atributo `att` começa com *val*
`E[att$="val"]`	Casa com qualquer elemento *E* cujo valor do atributo `att` termina com *val*
`E[att*="val"]`	Casa com qualquer elemento *E* cujo valor do atributo `att` contenha a substring *val*
`E F:nth-child(n)`	Casa com o elemento *F* que seja o n-th (enésimo) filho do elemento *E*
`E F:first-child`	Casa com os elementos *F* que sejam primeiro filho do elemento *E*
`E F:last-child`	Casa com os elementos *F* que sejam último filho do elemento *E*
`E F:only-child`	Casa com os elementos *F* que sejam único filho do elemento *E*
`E:enabled`	Casa com qualquer elemento *E* (por exemplo: um controle de formulário) que esteja habilitado
`E:disabled`	Casa com qualquer elemento *E* (por exemplo: um controle de formulário) que esteja desabilitado
`E:checked`	Casa com qualquer elemento *E* (por exemplo: um controle de formulário) que esteja marcado
`E:not(s)`	Casa com qualquer elemento *E* que não case com o seletor simples *s*

APÊNDICE B

Codificação de caracteres para HTML

Neste apêndice mostramos a codificação de caracteres e símbolos para HTML. As tabelas são constituídas de três colunas cada uma. Na primeira coluna é mostrado o caractere ou símbolo a ser apresentado, na segunda coluna a entidade nominal e na terceira coluna a entidade numérica que o representa.

Caracteres especiais para HTML

"	"	"	Ÿ	&Yulm;	Ÿ		‎	‎	”	”	”
&	&	&	ˆ	ˆ	ˆ		‏	‏	„	„	„
<	<	<	˜	˜	˜	–	–	–	†	†	†
>	>	>				—	—	—	‡	‡	‡
Œ	Œ	Œ				‘	‘	‘	‰	‰	‰
œ	œ	œ				’	’	’	‹	‹	‹
Š	Š	Š		‌	‌	‚	‚	‚	›	›	›
š	š	š		‍	‍	“	“	“	€	€	€

Caracteres matemáticos, gregos e símbolos para HTML

ƒ	ƒ	ƒ
Α	Α	Α
Β	Β	Β
Γ	Γ	Γ
Δ	Δ	Δ
Ε	Ε	Ε
Ζ	Ζ	Ζ
Η	Η	Η
Θ	Θ	Θ
Ι	Ι	Ι
Κ	Κ	Κ
Λ	Λ	Λ
Μ	Μ	Μ
Ν	Ν	Ν
Ξ	Ξ	Ξ
Ο	Ο	Ο
Π	Π	Π
Ρ	Ρ	Ρ
Σ	Σ	Σ
Τ	Τ	Τ
Υ	Υ	Υ
Φ	Φ	Φ
Χ	Χ	Χ
Ψ	Ψ	Ψ
Ω	Ω	Ω
α	α	α
β	β	β
γ	γ	γ
δ	δ	δ
ε	ε	ε
ζ	ζ	ζ

η	η	η
θ	θ	θ
ι	ι	ι
κ	κ	κ
λ	λ	λ
μ	μ	μ
ν	ν	ν
ξ	ξ	ξ
ο	ο	ο
π	π	π
ρ	ρ	ρ
ς	ς	ς
σ	σ	σ
τ	τ	τ
υ	υ	υ
φ	φ	φ
χ	χ	χ
ψ	ψ	ψ
ω	ω	ω
ϑ	ϑ	ϑ
ϒ	ϒ	ϒ
ϖ	ϖ	ϖ
■	•	•
…	…	…
′	′	′
″	″	″
‾	‾	‾
⁄	⁄	⁄
℘	℘	℘
ℑ	ℑ	ℑ
ℜ	ℜ	ℜ

™	™	™
ℵ	ℵ	ℵ
←	←	←
↑	↑	↑
→	→	→
↓	↓	↓
↔	↔	↔
↵	↵	↵
⇐	⇐	⇐
⇑	⇑	⇑
⇒	⇒	⇒
⇓	⇓	⇓
⇔	⇔	⇔
∀	∀	∀
∂	∂	∂
∃	∃	∃
∅	∅	∅
∇	∇	∇
∈	∈	∈
∉	∉	∉
∋	∋	∋
∏	∏	∏
∑	∑	∑
−	−	−
∗	∗	∗
√	√	√
∝	∝	∝
∞	∞	∞
∠	∠	∠
∧	∧	∧
∨	∨	∨

∩	∩	∩
∪	∪	∪
∫	∫	∫
⊥	⊥	⊥
⋅	⋅	⋅
⌈	⌈	⌈
⌉	⌉	⌉
∴	∴	∴
∼	∼	∼
≅	≅	≅
≈	≈	≈
≠	≠	≠
≡	≡	≡
≤	≤	≤
≥	≥	≥
⊂	⊂	⊂
⊃	⊃	⊃
⊄	⊄	⊄
⊆	⊆	⊆
⊇	⊇	⊇
⊕	⊕	⊕
⊗	⊗	⊗
⌊	⌊	⌊
⌋	⌋	⌋
〈	⟨	〈
〉	⟩	〉
◊	◊	◊
♠	♠	♠
♣	♣	♣
♥	♥	♥
♦	♦	♦

Caracteres para HTML – ISO-8859-1

			¸	¸	¸	Ð	Ð	Ð	è	è	è
¡	¡	¡	¹	¹	¹	Ñ	Ñ	Ñ	é	é	é
¢	¢	¢	º	º	º	Ò	Ò	Ò	ê	ê	ê
£	£	£	»	»	»	Ó	Ó	Ó	ë	&eulm;	ë
¤	¤	¤	¼	¼	¼	Ô	Ô	Ô	ì	ì	ì
¥	¥	¥	½	½	½	Õ	Õ	Õ	í	í	í
¦	¦	¦	¾	¾	¾	Ö	&Oulm;	Ö	î	î	î
§	§	§	¿	¿	¿	×	×	×	ï	&iulm;	ï
¨	¨	¨	À	À	À	Ø	Ø	Ø	ð	ð	ð
©	©	©	Á	Á	Á	Ù	Ù	Ù	ñ	ñ	ñ
ª	ª	ª	Â	Â	Â	Ú	Ú	Ú	ò	ò	ò
«	«	«	Ã	Ã	Ã	Û	Û	Û	ó	ó	ó
¬	¬	¬	Ä	Ä	Ä	Ü	&Uulm;	Ü	ô	ô	ô
			Å	Å	Å	Ý	Ý	Ý	õ	õ	õ
®	®	®	Æ	Æ	Æ	Þ	Þ	Þ	ö	&oulm;	ö
¯	¯	¯	Ç	Ç	Ç	ß	ß	ß	÷	&devide;	÷
°	°	°	È	È	È	à	à	à	ø	ø	ø
±	±	±	É	É	É	á	á	á	ù	ù	ù
²	²	²	Ê	Ê	Ê	â	â	â	ú	ú	ú
³	³	³	Ë	&Eulm;	Ë	ã	ã	ã	û	û	û
´	´	´	Ì	Ì	Ì	ä	&aulm;	ä	ü	&uulm;	ü
µ	µ	µ	Í	Í	Í	å	å	å	ý	ý	ý
¶	¶	¶	Î	Î	Î	æ	&aerlig;	æ	þ	þ	þ
■	·	·	Ï	&Iulm;	Ï	ç	ç	ç	ÿ	&yulm;	ÿ

APÊNDICE C

Elementos HTML

Esta tabela foi extraída do site do W3C e relaciona os elementos HTML. Fornece ainda informações sobre as tags de abertura e fechamento, elementos vazios, elementos em desuso e DTD a que se aplicam, além de uma breve descrição do elemento.

Nas colunas da tabela é adotada uma notação, ou legenda, para indicar restrições quando ao uso ou emprego do documento, como mostrado a seguir:

Notação	Significado
O	Uso opcional
P	Uso proibido
D	Elemento em desuso
V	Elemento vazio
L	DTD Loose
F	DTD Frameset

Elemento	Tag de abertura	Tag de fechamento	Vazio	Desuso	DTD	Descrição
A						âncora
ABBR						abreviaturas (p.ex: WWW, HTTP etc.)
ACRONYM						
ADDRESS						informações sobre o autor
APPLET				D	L	Java applet
AREA		P	V			área de um mapa de imagem no lado do cliente
B						estilo negrito

Elemento	Tag de abertura	Tag de fechamento	Vazio	Desuso	DTD	Descrição
BASE		P	V			URI base para o documento
BASEFONT		P	V	D	L	tamanho base para fonte
BDO						I18N BiDi over-ride
BIG						aumenta tamanho de fonte
BLOCKQUOTE						bloco de citação
BODY	O	O				seção corpo do documento
BR		P	V			quebra de linha
BUTTON						botão
CAPTION						título de tabela
CENTER				D	L	o mesmo que `DIV align=center`
CITE						citação
CODE						fragmento de código de computador
COL		P	V			coluna de tabela
COLGROUP		O				grupo de colunas de tabela
DD		O				item de lista de definição
DEL						texto apagado
DFN						instância de definição
DIR				D	L	lista de diretórios
DIV						container genérico, nivel de bloco
DL						lista de definição
DT		O				termo de lista de definição
EM						ênfase
FIELDSET						grupo de controles de formulário
FONT				D	L	altera estilo da fonte
FORM						formulário interativo
FRAME		P	V		F	janela secundária
FRAMESET					F	subdivisão de janelas
H1						título
H2						título
H3						título
H4						título
H5						heading

APÊNDICE D

Atributos HTML

Esta tabela foi extraída do site do W3C e relaciona os atributos HTML. Relaciona os elementos HTML aos quais se aplica cada um dos atributos e fornece informações sobre o tipo do atributo, a obrigatoriedade de uso, atributos em desuso e DTD a que se aplicam, além de um breve comentário sobre o atributo.

Nas colunas da tabela é adotada uma notação, ou legenda, para indicar restrições quando ao uso ou emprego do documento, como mostrado a seguir:

Notação	Significado
O	Uso obrigatório no elemento a que se aplica
D	Atributo em desuso
L	DTD Loose
F	DTD Framest

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
abbr	TD, TH	%Text;				abreviatura para texto da célula cabeçalho
accept-charset	FORM	%Charsets;				lista de caracteres suportados
accept	FORM, INPUT	%ContentTypes;				lista de tipos MIME para upload de arquivos
accesskey	A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA	%Character;				caractere para acesso por teclado
action	FORM	%URI;	O			local de processamento de formulário no servidor
align	CAPTION	%CAlign;		D	L	alinhamento para título de tabela

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
align	APPLET, IFRAME, IMG, INPUT, OBJECT	%IAlign;		D	L	alinhamento vertical ou horizontal
align	LEGEND	%LAlign;		D	L	alinhamento de legenda de fieldset
align	TABLE	%TAlign;		D	L	alinhamento de tabela na janela
align	HR	(left | center | right)		D	L	
align	DIV, H1, H2, H3, H4, H5, H6, P	(left | center | right | justify)		D	L	alinhamento de texto
align	COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR	(left | center | right | justify | char)				
alink	BODY	%Color;		D	L	cor do link selecionado
alt	APPLET	%Text;		D	L	breve descrição
alt	AREA, IMG	%Text;	O			breve descrição
alt	INPUT	CDATA				breve descrição
archive	APPLET	CDATA		D	L	lista de arquivos separados por vírgula
archive	OBJECT	CDATA				lista de URIs separados por espaço
axis	TD, TH	CDATA				lista de cabeçalhos relacionados separados por vírgula
background	BODY	%URI;		D	L	textura que se repete no fundo do documento
bgcolor	TABLE	%Color;		D	L	cor de fundo para tabela
bgcolor	TR	%Color;		D	L	cor de fundo para linhas
bgcolor	TD, TH	%Color;		D	L	cor de fundo para célula
bgcolor	BODY	%Color;		D	L	cor de fundo para o documento
border	TABLE	%Pixels;				espessura da borda em redor de tabela
border	IMG, OBJECT	%Pixels;		D	L	espessura da borda em link
cellpadding	TABLE	%Length;				espaçamento entre células
cellspacing	TABLE	%Length;				espaçamento entre células
char	COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR	%Character;				caractere de alinhamento, p.ex: char=':'
charoff	COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR	%Length;				offset para caractere de alinhamento

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
charset	A, LINK, SCRIPT	%Charset;				codificação de caratere do destino de link
checked	INPUT	(checked)				marcar por padrão botões radio e check boxes
cite	BLOCKQUOTE, Q	%URI;				URI para o documento de destino ou mensagem
cite	DEL, INS	%URI;				informações sobre as razões de mudança
class	todos os elementos, exceto BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, STYLE, TITLE	CDATA				lista de classes separadas por espaço
classid	OBJECT	%URI;				identifica uma implementação
clear	BR	(left | all | right | none)		D	L	controla o fluxo do texto
code	APPLET	CDATA		D	L	classe do arquivo para o elemento applet
codebase	OBJECT	%URI;				URI base para classid, data, arquivo
codebase	APPLET	%URI;		D	L	URI base opcional para applet
codetype	OBJECT	%ContentType;				tipo de conteúdo para o código
color	BASEFONT, FONT	%Color;		D	L	cor do texto
cols	FRAMESET	%MultiLengths;			F	lista de quantidade de colunas, padrão: 100% (1 coluna)
cols	TEXTAREA	NÚMERO	O			
colspan	TD, TH	NÚMERO				número de colunas transformadas em uma célula
compact	DIR, DL, MENU, OL, UL	(compacto)		D	L	redução de espaçamento
content	META	CDATA	O			informação associada
coords	AREA	%Coords;				lista de coordenadas separadas por vírgula
coords	A	%Coords;				para uso em mapa de imagens no lado do cliente
data	OBJECT	%URI;				dados do objeto
datetime	DEL, INS	%Datetime;				data-hora da modificação
declare	OBJECT	(declare)				declara flag para objeto
defer	SCRIPT	(defer)				UA pausam a inicialização do script

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
dir	todos os elementos, exceto APPLET, BASE, BASEFONT, BDO, BR, FRAME, FRAMESET, IFRAME, PARAM, SCRIPT	(ltr | rtl)				direção do texto
dir	BDO	(ltr | rtl)	O			direção
disabled	BUTTON, INPUT, OPTGROUP, OPTION, SELECT, TEXTAREA	(disabled)				desabilitado
enctype	FORM	%ContentType;				
face	BASEFONT, FONT	CDATA		D	L	lista de nomes de fontes separados por vírgula.
for	LABEL	IDREF				casa com ID de controle
frame	TABLE	%TFrame;				parte do frame a renderizar
frameborder	FRAME, IFRAME	(1 | 0)			F	existência de bordas em frames
headers	TD, TH	IDREFS				lista de id's para células cabeçalho
height	IFRAME	%Length;			L	altura
height	TD, TH	%Length;		D	L	altura
height	IMG, OBJECT	%Length;				altura
height	APPLET	%Length;	O	D	L	altura inicial
href	A, AREA, LINK	%URI;				URI para documento relacionado
href	BASE	%URI;				URI que atua como URI base
hreflang	A, LINK	%LanguageCode;				código de idioma
hspace	APPLET, IMG, OBJECT	%Pixels;		D	L	espaçamento horizontal
http-equiv	META	NAME				nome de cabeçalho HTTP
id	todos os elementos, exceto BASE, HEAD, HTML, META, SCRIPT, STYLE, TITLE	ID				identificador único
ismap	IMG, INPUT	(ismap)				mapa de imagem no lado do servidor
label	OPTION	%Text;				para uso em menus hierárquicos
label	OPTGROUP	%Text;	O			para uso em menus hierárquicos

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
lang	todos os elementos, exceto APPLET, BASE, BASEFONT, BR, FRAME, FRAMESET, IFRAME, PARAM, SCRIPT	%LanguageCode;				código de idioma
language	SCRIPT	CDATA		D	L	linguagem do script
link	BODY	%Color;		D	L	cor de links
longdesc	IMG	%URI;				link para descrição longa
longdesc	FRAME, IFRAME	%URI;			F	link para descrição longa
marginheight	FRAME, IFRAME	%Pixels;			F	altura da margem (em pixel)
marginwidth	FRAME, IFRAME	%Pixels;			F	largura de margem (em pixel)
maxlength	INPUT	NÚMERO				número máximo de caracteres em campos de texto
media	STYLE	%MediaDesc;				especifica a mídia a que se aplica
media	LINK	%MediaDesc;				especifica a mídia a que se aplica
method	FORM	(GET | POST)				método HTTP de envio de formulário
multiple	SELECT	(multiple)				padrão é seleção simples
name	BUTTON, TEXTAREA	CDATA				
name	APPLET	CDATA		D	L	permite identificação entre applets
name	SELECT	CDATA				nome de campo
name	FORM	CDATA				nome de formulário
name	FRAME, IFRAME	CDATA			F	nome do frame de destino
name	IMG	CDATA				nome de imagem
name	A	CDATA				nome de link destino
name	INPUT, OBJECT	CDATA				enviado com os dados do formulário
name	MAP	CDATA	O			para referenciar a um mapa
name	PARAM	CDATA	O			nome de propriedade
name	META	NAME				nome de metainformação
nohref	AREA	(nohref)				região sem função
noresize	FRAME	(noresize)			F	permissão para redimensionamento de frames

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
onkeyup	todos os elementos, exceto APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE	%Script;				uma tecla é solta
onload	FRAMESET	%Script;			F	todos os frames foram carregados
onload	BODY	%Script;				o documento foi carregado
onmousedown	todos os elementos, exceto APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE	%Script;				botão do dispositivo apontador é acionado
onmousemove	todos os elementos, exceto APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE	%Script;				apontador movido sobre o elemento
onmouseout	todos os elementos, exceto APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE	%Script;				apontador movido para fora do elemento
onmouseover	todos os elementos, exceto APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE	%Script;				apontador movido para o elemento

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
onmouseup	todos os elementos, exceto APPLET, BASE, BASEFONT, BDO, BR, FONT, FRAME, FRAMESET, HEAD, HTML, IFRAME, ISINDEX, META, PARAM, SCRIPT, STYLE, TITLE	%Script;				apontador solto
onreset	FORM	%Script;				formulário foi limpo
onselect	INPUT, TEXTAREA	%Script;				um texto foi selecionado
onsubmit	FORM	%Script;				formulário foi enviado
onunload	FRAMESET	%Script;			F	todos os frames foram removidos
onunload	BODY	%Script;				o documento foi removido
profile	HEAD	%URI;				informação de perfil
prompt	ISINDEX	%Text;		D	L	mensagem
readonly	TEXTAREA	(readonly)				
readonly	INPUT	(readonly)				somente leitura, para texto e senha
rel	A, LINK	%LinkTypes;				links relacionais
rev	A, LINK	%LinkTypes;				links reversos
rows	FRAMESET	%MultiLengths;			F	lista de quantidades, padrão: 100% (1 linha)
rows	TEXTAREA	NÚMERO	O			
rowspan	TD, TH	NÚMERO				número de linhas transformadas em uma célula
rules	TABLE	%TRules;				fios entre linhas e colunas
scheme	META	CDATA				seleção de conteúdos
scope	TD， TH	%Scope;				escopo para as células de cabeçalho
scrolling	FRAME, IFRAME	(yes | no | auto)			F	existência de barra de rolagem
selected	OPTION	(selected)				
shape	AREA	%Shape;				controle de interpretação por coordenadas
shape	A	%Shape;				para uso em mapa de imagem do lado do cliente
size	HR	%Pixels;		D	L	
size	FONT	CDATA		D	L	[+|-]n°. p.ex:. size="+1", size="4"

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
size	INPUT	CDATA				específico para cada tipo de campo
size	BASEFONT	CDATA	O	D	L	fonte base para o elemento FONT
size	SELECT	NÚMERO				número de linhas visíveis
span	COL	NÚMERO				atributo COL afetado por N colunas
span	COLGROUP	NÚMERO				número padrão de colunas no grupo
src	SCRIPT	%URI;				URI para script externo
src	INPUT	%URI;				para campos com imagens
src	FRAME, IFRAME	%URI;		F		endereço do conteúdo do frame
src	IMG	%URI;	O			URI da imagem a incorporar
standby	OBJECT	%Text;				mensagem a mostrar enquanto carrega
start	OL	NÚMERO		D	L	número inicial da sequência
style	todos os elementos, exceto BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, STYLE, TITLE	%StyleSheet;				informação sobre estilização
summary	TABLE	%Text;				propósito/estrutura para saída de sintetizador
tabindex	A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA	NÚMERO				posição na ordem de tabulação
target	A, AREA, BASE, FORM, LINK	%FrameTarget;			L	frame onde será renderizado
text	BODY	%Color;		D	L	cor dos textos no documento
title	todos os elementos, exceto BASE, BASEFONT, HEAD, HTML, META, PARAM, SCRIPT, TITLE	%Text;				título para o elemento
type	A, LINK	%ContentType;				tipo de conteúdo do elemento
type	OBJECT	%ContentType;				tipo de conteúdo para dados
type	PARAM	%ContentType;				tipo de conteúdo para valor, quando valuetype = ref
type	SCRIPT	%ContentType;	O			tipo de conteúdo da linguagem do script
type	STYLE	%ContentType;	O			tipo de conteúdo da linguagem de estilos

Atributo	Elementos a que se aplica	Tipo	Uso	Desuso	DTD	Comentário
type	INPUT	%InputType;	TEXT			tipo do controle input
type	LI	%LIStyle;		D	L	estilo do item de lista
type	OL	%OLStyle;		D	L	estilo númerico
type	UL	%ULStyle;		D	L	estilo marcador
type	BUTTON	(button | submit | reset)	submit			para uso em botões de formulário
usemap	IMG, INPUT, OBJECT	%URI;				usado em mapa de imagem no lado do cliente
valign	COL, COLGROUP, TBODY, TD, TFOOT, TH, THEAD, TR	(top | middle | bottom | baseline)				alinhamento vertical em células
value	INPUT	CDATA				para botão radio e checkboxes
value	OPTION	CDATA				valor padrão para escolha
value	PARAM	CDATA				valor da propriedade
value	BUTTON	CDATA				enviado ao servidor
value	LI	NÚMERO		D	L	limpa seqüência
valuetype	PARAM	(DATA | REF | OBJECT)	DATA			tipo de valor
version	HTML	CDATA	%HTML. Version;	D	L	constante
vlink	BODY	%Color;		D	L	cor de link visitado
vspace	APPLET, IMG, OBJECT	%Pixels;		D	L	espaçamento vertical
width	HR	%Length;		D	L	
width	IFRAME	%Length;			L	largura de frame
width	IMG, OBJECT	%Length;				largura do elemento
width	TABLE	%Length;				largura do elemento
width	TD, TH	%Length;		D	L	largura do elemento
width	APPLET	%Length;	O	D	L	largura inicial do elemento
width	COL	%MultiLength;				largura do elemento
width	COLGROUP	%MultiLength;				largura do elemento
width	PRE	NÚMERO		D	L	largura do elemento

APÊNDICE E

FAQ jQuery

Este apêndice é uma extensão da FAQ contida no site oficial da biblioteca jQuery. Alguns scripts que demonstram as respostas estão disponíveis no site do livro, no `arquivo-apA.1.html`.

Como selecionar um elemento usando seu atributo `id`?

Lembre-se de que o atributo `id` é um identificador único para um elemento, ou seja, somente um elemento, no documento, poderá conter o atributo `id` com um determinado valor. É claro que são admitidos vários ids, cada um com seu valor, em instâncias diferentes, no mesmo documento. Assim, a seleção usando o atributo `id` retornará um e somente um elemento.

Para selecionar um elemento ao qual se tenha atribuído o identificador `id` com valor *`idNome`*, use a sintaxe mostrada a seguir:

```
$('#idNome')
```

Como selecionar um elemento usando seu atributo `class`?

O atributo `class`, ao contrário do atributo `id`, pode ser usado quantas vezes forem necessárias em um mesmo elemento ou em elementos diferentes dentro de um documento.

Para selecionar os elementos aos quais se tenha atribuído o identificador `class` com valor *`classeNome`*, use a sintaxe mostrada a seguir:

```
$('.classeNome')
```

Um objeto selecionado via jQuery pode ser atribuído a uma variável JavaScript?

Sim, você pode atribuir a uma variável JavaScript um objeto jQuery e manipulá-lo normalmente com sintaxe jQuery, como mostrado no exemplo a seguir:

```
var meuDiv = $('#divNome');
var meuValor = $('#divNome').val();   // Armazena o valor do elemento
meuDiv.val('Olá mundo');              // Define o valor do elemento
```

Como verificar se existe um determinado elemento no documento?

Use a propriedade `length` para proceder à verificação como mostrado a seguir:

```
$('p').length();  // Retorna o número de parágrafos encontrados no documento
```

Lembre-se de que o uso de jQuery dispensa verificar a existência de um determinado objeto antes de aplicar-lhe um método. Observe o exemplo a seguir:

```
if ($('p').length())    // Verifica a existência de parágrafos
$('p').hide();          // Esconde os parágrafos caso existam
```

O teste condicional para verificar a existência de parágrafos é dispensável. Basta declarar-se `$('p').hide()` e, caso não existam parágrafos no documento, a instrução será ignorada sem gerar um erro.

Como verificar o estado de visibilidade de um determinado elemento no documento?

Para verificar o estado de visibilidade de um elemento, use os seletores `:visible` e `:hidden`.

Observe o exemplo a seguir:

```
var estadoVisivel = $('#idNome').is(':visible');
var estadoInvisivel = $('#idNome').is(':hidden');
```

Para aplicar um método em um elemento baseado na sua visibilidade, use a sintaxe mostrada a seguir:

```
$('#idNome:visible').css('color', 'red');
$('#idNome:hidden').show().animate({left:'+=200px'}, 'slow');
```

Como selecionar elementos cujo valor do `id` contenha caracteres especiais ?

Alguns frameworks geram valores para o atributo `id` contendo caracteres especiais, tais como colchetes ([..]) e ponto (.). Por exemplo: `<div id="id.Nome">` ou `<div id="id[Nome]">`

Em tais casos, você precisa usar uma sintaxe especial como a mostrada a seguir:

```
$('#id.Nome')        // Não funciona
$('#id\\.Nome')      // Funciona!
$('#id[Nome]')       // Não funciona
$('#id\\[Nome\\]')   // Funciona!
```

Como faço para habilitar e desabilitar um elemento?

Um elemento está desabilitado quando contém o atributo `disabled` com o valor `disabled`. Assim, para desabilitar um elemento, basta remover esse atributo e, para habilitá-lo, você deve configurar o atributo. O script é mostrado a seguir:

```
$('#xpto').attr('disabled', 'disabled');  // Desabilita o elemento cujo id tem o valor xpto
$('#xpto').removeAttr('disabled');  // Habilita o elemento cujo id tem o valor xpto
```

Desenvolva uma página, para demonstrar o funcionamento do script mostrado, com a seguinte marcação HTML. Não se esqueça de linkar a página à biblioteca.

```
<select id="xpto" style="width:200px">
<option>Opção um</option>
<option>Opção dois</option>
</select>
<input type="button" value="Desabilitar" onclick="$('#xpto').attr('disabled',
    'disabled')" />
<input type="button" value="Habilitar" onclick="$('#xpto').removeAttr('disabled')" />
```

Como faço para marcar e desmarcar um elemento `input`?

Um elemento está marcado quando contém o atributo `checked` com o valor `checked`. Assim, para marcar um elemento, basta remover esse atributo e, para desmarcá-lo, você deve configurar o atributo. O script é mostrado a seguir:

```
$('#ypto').attr('checked', 'checked'); // Marca o elemento cujo id tem o valor ypto
$('#ypto').attr('checked', ' ');    // Desmarca o elemento cujo id tem o valor ypto
```

Desenvolva uma página, para demonstrar o funcionamento do script mostrado, com a seguinte marcação HTML. Não se esqueça de linkar a página à biblioteca.

```
<label><input type="checkbox" id="ypto" />Marca/Desmarca</label>
<input type="button" value="Marcar" onclick="$('#ypto').attr('checked', 'checked')" />
<input type="button" value="Desmarcar" onclick="$('#ypto').attr('checked', '')" />
```

Na sintaxe `$('ypto').attr('checked', '')`, não use espaço em branco no valor do atributo ao declará-lo vazio. Essa sintaxe equivale a `$('ypto').removeAttr('checked')`.

Como obter o valor textual da opção selecionada em um elemento `select`?

Elementos `option` de um `select` possuem dois valores distintos: o valor do atributo `value` e o valor textual do elemento. Desenvolva uma página, para demonstrar o funcionamento do script mostrado, com a seguinte marcação HTML. Não se esqueça de linkar a página à biblioteca.

```
<select id="meuSelect" style="width:200px">
<option value="fruta_um">Abacate</option>
<option value="fruta_dois">Abacaxi</option>
<option value="fruta_tres">Banana </option>
<option value="fruta_quatro">Figo</option>
<option value="fruta_cinco">Goiaba</option>
<option value="fruta_seis">Laranja</option>
</select>
<input type="button" value="Valor" onclick="alert($('#meuSelect').val())" />
<input type="button" value="Texto" onclick="alert($('#meuSelect option:selected').
   text())" />
```

Como substituir o texto do terceiro elemento de um conjunto de seis elementos?

Tanto o seletor `:eq()` como o método `eq()` permitem fazer a seleção. Desenvolva uma página, para demonstrar o funcionamento do script, com a seguinte marcação HTML. Não se esqueça de linkar a página à biblioteca.

▶ **HTML**:

```
<ol id="minhaLista">
   <li>texto do primeiro item</li>
   <li>texto do segundo item</li>
   <li>texto do terceiro item</li>
   <li>texto do quarto item</li>
   <li>texto do quinto item</li>
   <li>texto do sexto item</li>
</ol>
```

▶ **jQuery**:

```
// Não funciona
$('#minhaLista').find('li').eq(2).text().replace('texto do terceiro item','TEXTO ALTERADO');
```

```
// Funciona
   var $elementoTres = $('#minhaLista').find('li').eq(2);
   var textoElementoTres = $elementoTres.text().replace('texto do terceiro item',
      'TEXTO ALTERADO');
  $elementoTres.text(textoElementoTres).css('color', 'red');
```

O primeiro script mostrado não funciona porque o objeto retornado na cadeia é o texto substituto para o terceiro elemento. Isto porque `replace()` é um método JavaScript de manipulação de strings e não um método jQuery. Observe, no segundo script, que se armazenou o texto substituto em uma variável denominada `textoElementoTres` e com o uso do método `text('texto a inserir')` se efetuou a substituição do texto.

Por que a animação de um elemento faz que se comporte como elemento nível de bloco?

Ao se aplicar um efeito de animação que altere as propriedades CSS `height` e/ou `width`, tais como os efeitos show, hide, slideUp ou slideDown, o objeto animado, durante o curso da animação, tem sua propriedade CSS `display` configurada para `block`, porque `height` e `width` são propriedades aplicáveis a elementos nível de bloco. Ao término da animação, a propriedade `display` é configurada para seu valor original.

Desenvolva uma página, para visualizar esse comportamento, animando um elemento in-line como mostrado na marcação HTML a seguir.

```
...
<style type="text/css" media="all">
em {background:#ff6;}
</style>
<script type="text/javascript" src="../jquery-1.2.6.js"></script>
<script type="text/javascript">
   $(document).ready(function() {
      $('em').click(function() {
         $(this).hide(3000);
      });
   });
</script>
</head>
<body>
<em>Lorem ipsum dolor</em>
</body>
</html>
```

Qual é a diferença entre `append()` e `appendTo()`?

Estes dois métodos destinam-se a inserir conteúdos dentro de um elemento HTML. A inserção será imediatamente antes da tag de fechamento do elemento e o conteúdo a ser inserido pode ser uma string, um trecho de marcação HTML ou um objeto jQuery retirado do próprio documento.

A única diferença entre os dois métodos é a sintaxe, como mostrado a seguir:

```
$('p').append('<b>texto negrito</b>');
$('<b>texto negrito</b>').appendTo('p');
```

Qual é a diferença entre `prepend()` e `prependTo()`?

Estes dois métodos destinam-se a inserir conteúdos dentro de um elemento HTML. A inserção será imediatamente após a tag de abertura do elemento e o conteúdo a ser inserido pode ser uma string, um trecho de marcação HTML ou um objeto jQuery retirado do próprio documento.

A única diferença entre os dois métodos é a sintaxe, como mostrado a seguir:

```
$('p').prepend('<b>texto negrito</b>');
$('<b>texto negrito</b>').prependTo('p');
```

Qual é a diferença entre `after()` e `insertAfter()`?

Estes dois métodos destinam-se a inserir conteúdos imediatamente após um elemento HTML. O conteúdo a inserir pode ser uma string, um trecho de marcação HTML ou um objeto jQuery retirado do próprio documento.

A única diferença entre os dois métodos é a sintaxe, como mostrado a seguir:

```
$('p').after('<b>texto negrito</b>');
$('<b>texto negrito</b>').insertAfter('p');
```

Qual é a diferença entre `before()` e `insertBefore()`?

Estes dois métodos destinam-se a inserir conteúdos imediatamente antes de um elemento HTML. O conteúdo a inserir pode ser uma string, um trecho de marcação HTML ou um objeto jQuery retirado do próprio documento.

A única diferença entre os dois métodos é a sintaxe, como mostrado a seguir:

```
$('p').before('<b>texto negrito</b>');
$('<b>texto negrito</b>').insertBefore('p');
```

Qual o comportamento dos métodos de inserção quando o conteúdo a inserir é um objeto jQuery retirado do documento?

Os métodos de inserção `append()`, `appendTo()`, `prepend()`, `prependTo()`, `after()`, `insertAfter()`, `before()` e `insertBefore()` podem receber como parâmetros de inserção objetos jQuery retirados do próprio documento. Neste caso, a inserção se processa de duas maneiras diferentes, dependendo da quantidade de alvos da inserção. Veja um exemplo prático para esclarecer esses comportamentos.

Observe os códigos a seguir.

▶ **HTML**:

```
...
<body>
    <b>Texto negrito</b>
    <p>Primeiro parágrafo: </p>
    <p>Segundo parágrafo: </p>
</body>
...
```

▶ **jQuery**:

```
<script type="text/javascript">
    $(document).ready(function() {

        $('p').after($('b'));
    });
</script>
```

O script jQuery proposto insere dentro e no final dos dois parágrafos do documento a marcação e conteúdo do elemento `b`. Assim, o resultado final é o mostrado a seguir:

▶ **HTML**:

```
...
<body>
    <b>Texto negrito</b>
    <p>Primeiro parágrafo: <b>Texto negrito</b></p>
    <p>Segundo parágrafo: <b>Texto negrito</b></p>
</body>
...
```

Note que os dois parágrafos receberam uma *cópia* do elemento `b`.

Qual a diferença entre os métodos de inserção e o método `html(val)` ?

Tal como os métodos de inserção, o método `html(val)` destina-se a inserir conteúdos, admite os mesmos parâmetros e comporta-se semelhantemente a esses métodos. A diferença é que o método `html(val)` se destina a inserir conteúdos em elementos vazios. Caso você use esse método para inserir conteúdos em um elemento não vazio, tais conteúdos serão retirados e a inserção será feita normalmente, como se o elemento estivesse vazio.

Observe a seguir a sintaxe de emprego deste método:

```
$('div.classeNome'). html('<p>Parágrafo inserido</p>');
```

Insere um parágrafo nos divs cujo atributo *classe* tenha o valor *classeNome*.

Qual é diferença entre os métodos `empty()` e `remove()`?

O método `empty()` retira os conteúdos e mantém o elemento no DOM e o método `remove()` retira o elemento do DOM. É importante ressaltar que o método `remove()` preserva o objeto jQuery pronto para uso posterior.

Observe o exemplo mostrado a seguir:

```
$('p#idNome').remove().appendTo('body');
```

Retira do DOM o elemento parágrafo cujo atributo `id` é *idNome* e insere-o, assim como seus conteúdos, no elemento *body*. Caso tivesse usado o método `empty()`, a inserção teria sido de um parágrafo vazio.

Como construir uma chave de mudança de estilo?

Esta é uma funcionalidade em desenvolvimento de sites encontrada com bastante frequência. Ao usuário se fornece a opção de navegar no site escolhendo um tema de sua preferência entre alguns disponibilizados pelo desenvolvedor do site. Não só mudanças no visual do site, mas também opção de temas para navegar com apresentação em alto contraste, modo de impressão ou fontes aumentadas.

Observe os códigos a seguir:

Qual a diferença entre os métodos de inserção e o método html(val) ?

Tal como os métodos de inserção, o método html(val) destina-se a inserir conteúdos, admite os mesmos parâmetros e comporta-se semelhantemente a esses métodos. A diferença é que o método html(val) se destina a inserir conteúdos em elementos vazios. Caso você use esse método para inserir conteúdos em um elemento não vazio, tais conteúdos serão retirados e a inserção será feita normalmente, como se o elemento estivesse vazio.

Observe a seguir a sintaxe de emprego deste método:

```
$('div.classeNome'). html('<p>Parágrafo inserido</p>');
```

Insere um parágrafo nos divs cujo atributo *classe* tenha o valor *classeNome*.

Qual é diferença entre os métodos empty() e remove()?

O método empty() retira os conteúdos e mantém o elemento no DOM e o método remove() retira o elemento do DOM. É importante ressaltar que o método remove() preserva o objeto jQuery pronto para uso posterior.

Observe o exemplo mostrado a seguir:

```
$('p#idNome').remove().appendTo('body');
```

Retira do DOM o elemento parágrafo cujo atributo id é *idNome* e insere-o, assim como seus conteúdos, no elemento *body*. Caso tivesse usado o método empty(), a inserção teria sido de um parágrafo vazio.

Como construir uma chave de mudança de estilo?

Esta é uma funcionalidade em desenvolvimento de sites encontrada com bastante frequência. Ao usuário se fornece a opção de navegar no site escolhendo um tema de sua preferência entre alguns disponibilizados pelo desenvolvedor do site. Não só mudanças no visual do site, mas também opção de temas para navegar com apresentação em alto contraste, modo de impressão ou fontes aumentadas.

Observe os códigos a seguir:

Qual a diferença entre os métodos de inserção e o método `html(val)` ?

Tal como os métodos de inserção, o método `html(val)` destina-se a inserir conteúdos, admite os mesmos parâmetros e comporta-se semelhantemente a esses métodos. A diferença é que o método `html(val)` se destina a inserir conteúdos em elementos vazios. Caso você use esse método para inserir conteúdos em um elemento não vazio, tais conteúdos serão retirados e a inserção será feita normalmente, como se o elemento estivesse vazio.

Observe a seguir a sintaxe de emprego deste método:

```
$('div.classeNome'). html('<p>Parágrafo inserido</p>');
```

Insere um parágrafo nos divs cujo atributo *classe* tenha o valor *classeNome*.

Qual é diferença entre os métodos `empty()` e `remove()`?

O método `empty()` retira os conteúdos e mantém o elemento no DOM e o método `remove()` retira o elemento do DOM. É importante ressaltar que o método `remove()` preserva o objeto jQuery pronto para uso posterior.

Observe o exemplo mostrado a seguir:

```
$('p#idNome').remove().appendTo('body');
```

Retira do DOM o elemento parágrafo cujo atributo `id` é *idNome* e insere-o, assim como seus conteúdos, no elemento *body*. Caso tivesse usado o método `empty()`, a inserção teria sido de um parágrafo vazio.

Como construir uma chave de mudança de estilo?

Esta é uma funcionalidade em desenvolvimento de sites encontrada com bastante frequência. Ao usuário se fornece a opção de navegar no site escolhendo um tema de sua preferência entre alguns disponibilizados pelo desenvolvedor do site. Não só mudanças no visual do site, mas também opção de temas para navegar com apresentação em alto contraste, modo de impressão ou fontes aumentadas.

Observe os códigos a seguir:

A

Agrupamento de seletores, 59
Alerta, 17
Ampliação de imagem, 335
Animação
  com fadeIn() e fadeOut(), 233
  com show() e hide(), 215
  com slideUp() e slideDown(), 231
  com toggle(), 219
  personalizada com animate(), 234
Animações básicas, 213
Apresentação de tabelas, 275
Arquivos para download, 19
Árvore do documento, 123, 392

B

Box Model, 189

C

Carrosséis horizontal e vertical, 360
Carrossel
  com comandos externos, 365
  com efeito thickbox, 368
  com scroll automático, 362
Cascading Style Sheets.*ver* CSS
Conflitos com outras bibliotecas, 43
Construtor jQuery, 40
Convenções tipográficas, 17
Coordenadas, 127

D

Desabilitar campos, 312
Destacar
  colunas, 292
  linhas, 291
  linhas seletivamente, 295
Destaques em geral, 20
DOCTYPE, 214

E

Efeitos
  básicos, 173
  corrediços, 177
  de opacidade, 179
  lightbox, 335
  para destacar, 290
  personalizados, 181
  zebra, 280
Efeito zebra em tabelas.*Ver* Zebra
Elemento
  adjacente, 57
  legend, 318
Elemento-filho, 57
Elemento-irmão, 57
  adjacente, 57
Elemento-pai, 57
Elemento-raiz, 214
Encadeamento, 42
Estilização geral, 123
event.pageX/Y, 170
event.preventDefault(), 170
event.stopPropagation(), 171
event.target, 170
event.type, 169
Eventos, 151
  auxiliares, 151
  de interação, 164
eXtensible Markup Language.*ver* XML

F

FAQ CSS, 239
Filtros
  básicos, 64
  para formulários, 97
  para seletores-filho, 85
Flags para agentes de usuário, 186
Folha de estilo, 214
Formulários, 307
Funções
  utilitárias, 42
  utilitárias personalizadas, 198
Funções-padrão, 45

G

Galeria com thumbnails, 342
Galerias de imagens, 340

I

Imagens, 325
  Atributo alt, 326
  Atributo longdesc, 327
  decorativas, 326

Inspeção do DOM, 136
Instalar jQuery, 30
Interatividade, 27

J

jQuery.boxModel, 188
jQuery.browser, 186
jQuery/UI, 356

M

Manipulação
  de conteúdos, 113
  de conteúdos html, 108
  de textos, 109
  de valores, 111
Manipuladores de eventos, 166
Marcação, 18
  de tabelas, 275
Menu Maujor, 379
Menu sanfona, 375
Métodos
  $(callback), 48
  $(html), 47
  add(), 139
  addClass(), 105
  after(), 116
  andSelf(), 147
  animate(), 331
  animate(definições [velocidade] [aceleração] [função]), 181
  append(), 113
  appendTo(), 115
  attr({atributo:valor}), 102
  attr(atributo, valor), 103
  attr(nome_atributo), 101
  before(), 117
  bind(), 166
  blur(), 151
  change(), 152
  children(), 140
  click(), 153
  clone(), 122
  clone(true), 122
  css({propriedade:'valor', propriedade:'valor', ...}), 125
  css('propriedade', 'valor'), 124
  css(propriedade), 123
  dblclick(), 153
  each(callback), 49
  each(objeto função(chave ou índice valor)), 190
  empty(), 121
  end(), 146
  eq(), 50
  error(), 154
  fadeIn(), 179
  fadeOut(), 179
  fadeTo(), 180
  filter(filtro), 136
  filter(função), 137
  find(), 141
  focus(), 154
  get(), 51
  get(índice), 52
  grep(), 192
  hasClass(), 106
  height(), 132
  height(valor), 133
  hide(), 174
  hover(), 165
  html(), 108
  html(valor), 109
  inArray(), 195
  index(), 52
  innerHeight(), 135
  innerWidth(), 135
  insertAfter(), 116
  insertBefore(), 117
  is(), 137
  isFunction(), 196
  keydown(), 155
  keypress(), 156
  keyup(), 156
  length, 50
  load(), 156
  makeArray(), 193
  map(array função(valor índice)), 193
  map(função), 149
  mousedown(), 156
  mousemove(), 160
  mouseout(), 158
  mouseover(), 158
  mouseup(), 157
  next(), 145
  nextAll(), 145
  noConflict(), 53
  not(), 138
  offset(), 125
  one(), 167
  outerHeight(), 135

outerWidth(), 133
parent([expressão]), 142
parents([expressão]), 143
position(), 126
prepend(), 116
prependTo(), 116
prev(), 144
prevAll(), 145
ready(), 38
remove(), 120
removeAttr(), 104
removeClass(), 106
replaceAll(), 121
replaceWith(), 121
resize(), 160
scroll(), 161
scrollLeft(), 129
scrollLeft(valor), 129
scrollTop(), 129
select(), 162
show(), 173
siblings(), 148
slice(), 138
slideDown(), 177
slideToggle(), 178
slideUp(), 177
stop(), 183
submit(), 162
text(), 109
text(valor), 110
toggle(), 175
toggle(f1 f2 [f3...fN]), 164
toggleClass(), 107
trigger(), 168
trim(), 197
unbind(), 169
unique(), 196
unload(), 163
val(), 111
val(valor_atributo_value), 112
width(), 130
width(valor), 131
wrap(elemento), 118
wrap(html), 117
wrapAll(elemento), 119
wrapAll(html), 118
wrapInner(elemento), 120
wrapInner(html), 120
Mover e copiar, 115

N

Notas sobre eventos, 169

O

Opacidade, 333
Operação com string, 197
Operações com arrays e objetos, 190

P

Padrões Web, 28
Página de notícias, 265
Placeholder para campos, 307
Plug-ins
de terceiros, 355
jCarousel, 356
jQuery Accordion, 370
nativos, 356
Posicionamento, 125
Progressão aritmética, 286
Pseudoseletores
:animated, 73
:button, 95
:checkbox, 92
:checked, 98
:contains(texto), 73
:disabled, 97
:empty, 74
:enabled, 97
:eq(índice), 69
:even, 67
:file, 95
:first, 64
:first-child, 85
:gt(índice), 70
:has(seletor2), 75
:hidden, 76, 96
:image, 94
:input, 88
:last, 65
:last-child, 86
:lt(índice), 71
:not(seletor2), 66
:nth-child(índice/even/odd/equação), 87
:odd, 68
:only-child, 86
:parent, 75
:password, 90
:radio, 91
:reset, 93

:selected, 99
:submit, 92
:text, 89
:visible, 78

R

Resig, John, 26
Revelar
campos, 316
conteúdos, 239
e esconder linhas, 300

S

Sanfona simples, 372
Seletores
$(:header), 72
$(ancestral descendente), 60
$(anterior + próximo), 62
$(anterior ~ irmãos), 63
$(classe), 58
$(elemento), 58
$(elementos), 47
$(expressão, [contexto]), 46
$(id), 57
$(pai > filho), 61
$(seletor1, seletor2, seletor3, ...), 59
atributo, 391
avançados, 396
classe, 390
composto, 60, 392
CSS, 41
de atributo, 79
descendente, 394
de visibilidade, 76
identificador único, 390
jQuery, 55
para formulários, 88
Seletor-filho, 394
Seletor-irmão adjacente, 395
seletor[atributo != "valor"], 81
seletor[atributo $= "valor"], 82
seletor[atributo *= "valor"], 83
seletor[atributo = "valor"], 80
seletor[atributo], 79
seletor[atributo ^= "valor"], 82
simples, 57, 390
tipo, 389
universal, 391
Site do livro, 21
Slide-show, 347
Suavizando a animação, 228

T

Tabela de horários de ônibus, 276
Terminologia, 18
thickbox, 368

U

Utilitárias disponíveis, 185

V

Validação de formulários, 307
Validar, 322
Variação de opacidade, 333

W

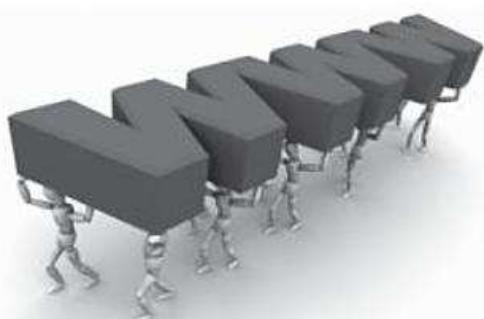
window.onload, 36

Z

Zebra
dois-dois, 283
par-ímpar, 281
três-três, 288
três cores, 286

DO MESMO AUTOR

Construindo Sites com CSS e (X)HTML
ISBN: 978-85-7522-139-6
Formato: 17 x 24 cm
Páginas: 448
Ano: 2007
Preço: R$ 75,00

Criando Sites com HTML
ISBN: 978-85-7522-166-2
Formato: 17 x 24 cm
Páginas: 432
Ano: 2008
Preço: R$ 75,00

www.novatec.com.br